Pós-Graduação em Ciência da Computação

VEPersonal – Uma Infra-Estrutura para Geração e Manutenção de Ambientes Virtuais Adaptativos

Por

MARCUS SALERNO DE AQUINO

Tese de Doutorado

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

RECIFE, DEZEMBRO/2007

Universidade Federal de Pernambuco

CENTRO DE INFORMÁTICA

PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

Marcus Salerno de Aquino

VEPersonal – Uma Infra-Estrutura para Geração e Manutenção de Ambientes Virtuais Adaptativos

Este Trabalho foi apresentado à pós-graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de doutor em Ciência da Computação.

Orientador: Prof. Dr. Alejandro C. Frery

Co-orientador: Prof. Dr. Fernando da Fonseca de Souza

RECIFE, DEZEMBRO/2007

Aquino, Marcus Salerno de

VEPersonal – uma infra-estrutura para geração e manutenção de ambientes virtuais adaptativos / Marcus Salerno de Aquino – Recife : O Autor, 2007.

xiv, 170 folhas : il., fig., quadro.

Tese (doutorado) - Universidade Federal de Pernambuco. CIn. Ciência da computação, 2007.

Inclui bibliografia e apêndice.

1. Realidade virtual 2. Adaptatividade. 3. Sistema multi-agentes. 4. X3D. 5. Perfil do usuário. I. Título.

006.8 CDD (22.ed.) MEI2008-17

Ao meu sogro (in-memoriam), pelas conversas, histórias e casos contados por ele todo final de tarde e que muitas vezes tive que abdicar para poder concluir esta tese.

Agradecimentos

Gostaria de agradecer inicialmente a Deus por me dar sabedoria nas horas difíceis e me iluminar na conclusão deste trabalho. Ele também foi o responsável por me ajudar a escalar os degraus desta tese, avançando um degrau a cada dia, mesmo que muitas vezes os degraus parecessem intermináveis.

Em seguida, gostaria de agradecer aos meus dois orientadores. Digo isso, por reconhecer em ambos a mesma competência, infinita paciência no acompanhamento deste trabalho e dignos do meu reconhecimento.

Ao meu orientador Prof. Alejandro Frery por me aceitar como seu aluno no programa de doutorado do CIn/UFPE e acreditar na minha proposta e no meu trabalho. A ele também quero agradecer as palavras de incentivo e de apoio na elaboração desta tese e nas publicações que se sucederam.

Ao meu orientador Prof. Fernando Fonseca que também me acolheu como seu aluno e, durante toda a etapa desta tese, me fez pensar, questionar e refletir dentro de uma visão holística sobre o tema de pesquisa escolhido. Ele ainda debruçou-se com dedicação e perspicácia na correção e revisão deste trabalho e dos artigos publicados, sendo sempre muito preciso nas suas observações e questionamentos. Quero também registrar a gratidão de poder encontrar em Prof. Fernando um amigo e acima de tudo um pesquisador de extrema competência e de grande visão científica, que sempre me indicou os caminhos a trilhar de forma a poder solucionar as dificuldades enfrentadas durante as etapas desta pesquisa.

Agradecimento especial aos alunos que participaram, de forma voluntária, do projeto VEPersonal. Em particular a Daniel Abella, pela sua competência, dedicação, profissionalismo e pelas soluções encontradas na integração das diversas tecnologias utilizadas neste trabalho. A Rodrigo Fujioka que também teve uma participação fundamental no desenvolvimento do projeto, pela sua dedicação, principalmente na implantação do SGBD, elaboração do ambiente Web e integração das tecnologias. A Marcelo Vieira que, apesar de ter sido engajado no final do projeto, deu sua contribuição durante a fase de testes.

Ao aluno de Iniciação Científica Emannuel Gomes pela ajuda nos primeiros passos da concepção da proposta de adaptatividade utilizada no VEPersonal, e aos alunos de graduação Marcus Vinícius, Domingos, Rodrigo Almeida, Maycon e Adson, que em seus Trabalhos de Graduação também tiveram participação na construção do VEPersonal.

À minha esposa Dêlma, companheira de todos os momentos, que sempre me incentivou e me apoiou nesta jornada, e que muitas vezes abdicou do seu trabalho e do seu conforto para que eu pudesse me dedicar a esta tese. A ela eu reconheço meu amor e gratidão. Tenho certeza que sem ela eu não estaria escrevendo todos os agradecimentos pela conclusão deste trabalho.

Às minhas filhas Camille e Caroline que souberam entender, em diversos momentos, a minha ausência e a minha falta de paciência. Com certeza, todo esse esforço foi para o bem delas.

Agradeço aos meus pais, Helena e Regis, que apesar da distância, sempre me incentivaram com palavras de apoio nos incansáveis telefonemas dominicais.

Aos meus sogros, Carminha e Carloto (in-memoriam) que, pela proximidade, estiveram sempre presentes no nosso dia-a-dia e na nossa convivência, sempre me incentivando na conclusão desta tese.

Aos colegas do Departamento de Sistemas e Computação da UFCG que acreditaram no meu trabalho ao me liberarem para o doutorado e "carregaram o piano" durante os quatro anos de minha ausência.

Por fim, um agradecimento muito especial a Mel, minha cachorra *poodle*, que durante as infindáveis horas de estudo que passei no computador ela ficou ao meu lado, como se quisesse compartilhar os prazerosos momentos de aprendizagem. Por isso, dedico esta tese também a ela dizendo: "E aí Mel, vamos estudar?".

Resumo

Os avanços das técnicas de Realidade Virtual têm contribuído para que os Ambientes Virtuais se tornem mais dinâmicos e mais realistas. Além disso, essas técnicas são importantes para tornar tais sistemas mais próximos da realidade dos usuários. Técnicas para geração de ambientes virtuais que se adaptam ao perfil do usuário têm sido desenvolvidas, incorporando procedimentos de acompanhamento de ações e modificando o ambiente em função deste comportamento. Esta tese apresenta o sistema VEPersonal (*Personalized Virtual Environment*), uma infra-estrutura para gerenciamento de componentes de realidade virtual, que permite adaptar ambientes virtuais, em tempo real, em função do nível de conhecimento do usuário ou de suas necessidades. Tais componentes são armazenados em um SGBD com suporte a XML para permitir a reutilização na construção de novos mundos virtuais. Cada objeto do mundo pode conter vários níveis de informação que são apresentados ao usuário de acordo com o seu nível de conhecimento ou a sua capacidade de aprendizado. Uma sociedade de agentes monitora as ações do usuário, calcula as atualizações necessárias e determina quais objetos e qual o nível de detalhes devem ser apresentados ao mundo virtual do usuário corrente. Um modelo do usuário e um modelo do ambiente mantêm informações atualizadas sobre o perfil do usuário e o ambiente utilizado, respectivamente, facilitando a tomada de decisão por parte dos agentes. A interface entre o servidor e a máquina do cliente utiliza uma comunicação síncrona que permite gerenciar as ações do usuário e atualizar o seu ambiente em tempo real. Com o VEPersonal, ambientes virtuais adaptativos para a Web podem ser gerados de acordo com a evolução cognitiva do usuário, de maneira dinâmica e personalizada.

Palavras chaves: ambientes virtuais, adaptatividade, sistemas multi-agentes, X3D, perfil do usuário, Xj3D

Abstract

The advances of Virtual Reality techniques have aided Virtual Environments to become more dynamic and more realistic. Furthermore, it is of paramount importance to turn such systems closer to users' reality. This has led to the need of environments that can both track user's actions and be able of adapting themselves to the users' profile. This thesis presents the VEPersonal system (Personalized Virtual Environment) an infrastructure for managing virtual reality components that allows the personalization of virtual environments in real time in accordance to the user's profile. Such components are stored in a DBMS with XML support, in order to improve their reuse for constructing new virtual worlds. Each of such objects can contain several levels of information and one of them will be introduced to the user in agreement with his/her learning capacity and/or knowledge level. An agents' society monitors user's actions, and calculates the necessary updates to the virtual word and determines which objects and which detail level should be presented. A user's model and an environment model keep information up-to-date about the user profile and the environment respectively, facilitating the agents' decisions. The interface between the server and the client machine uses a synchronous communication that allows to manage user's actions and to update his or her environment in real time. With this proposal an adaptive virtual environment for the Web can be generated which will follow the user's cognitive evolution in a dynamic and personalized way

Key words: virtual environments, adaptativiy, multiagent systems, X3D, user profile, Xj3D

SUMÁRIO

Lista de Figuras

Lista de Quadros

Lista de Siglas

ACL	- Agent Communication Language
AdapTIVE	- Adaptive Threedimensional Intelligent and Virtual Environment
ADVIRT	- The Adaptive VR Store
AM	- Aprendizagem de Máquina
API	- Application Program Interface
AV	- Ambientes Virtuais
AVI	- Ambientes Virtuais Inteligentes
AWE3D	- Adaptive WEb 3D
BC	- Base de Conhecimentos
CLOB	- Character Large Object
CVA	- Companheiro Virtual de Aprendizado
DAO	- Data Access Object
Data Map	- mapa de dados
Design patterns	- padrões de projeto
DTD	- Document Type Definition
FC	- Fatores de Confiança
FC_{user}	- FC do usuário
IRE	- Identifying Referential Expressions
JADE	- Java Agent DEvelopment framwork
JDBC	- Java Database Connectivity
JVM	- Java Virtual Machine
JWS	- Java Web Start
KQML	- Knowledge Query and Manipulation Language
LOD	- Levels of Details
MA	- Modelo do Ambiente
Mapping Functions	- Funções de mapeamento
MBL	- Modelos Baseados em Lógica
MDP	- Métodos de Definição de Perfil
MU	- Modelo do Usuário
NURBS	- Non-uniform Rational B-Splines
PMK	- Project Management Knowledge Learning Environment

POM	- Project Object Model
POSIX	- Portable Operating System Interface
Query Functions	- Linguagens de consultas
RMI	- Remote Method Invocation
RP	- Reconhecimento de Planos
RV	- Realidade Virtual
SAI	- Scene Access Interface
SGBD	- Sistema de Gerenciamento de Banco de Dados
SL	- Semantic Language
SMA	- Sistema Multi-Agentes
TCP/IP	- Transmission Control Protocol / Internet Protocol
TFIDF	- Term Frequency Inverse Document Frequency
Trigger Functions	- Gatilhos
VEPersonal	- Personalized Virtual Environment
VRML	- Virtual Reality Modeling Language
VICTOR	- Virtual Intelligent Companion for TutOring and Reflection
XQL	- XML Query Language
XSL	- Extensible Stylesheet Language
XUpdate	- XML Update Language
X3D	- Extensible 3D Language

1. Introdução

Ambientes Virtuais (AV) são ambientes interativos, compostos por objetos tridimensionais e gerados em tempo real por um sistema computacional. O objetivo de tais ambientes é simular um ambiente real ou construir ambientes imaginários, permitindo a um ou mais usuários, interagirem através da visualização e manipulação de objetos. Essa interação possibilita ao usuário um aumento do sentimento de presença no ambiente [Kirner e Siscoutto, 2007]. Todas as abordagens sobre AV, discutidas nesta tese, tratam de ambientes tridimensionais, compostos por objetos 3D, dinâmicos e interativos.

Nos AV, a interação mais simples é a navegação, decorrente da movimentação do usuário no espaço tridimensional, resultando na visualização de novos pontos de vista do cenário. Nesse caso, não há mudanças no AV, mas somente um passeio exploratório. Interações, propriamente ditas, com alterações no ambiente virtual, ocorrem quando o usuário entra no espaço virtual das aplicações e visualiza, explora, manipula e aciona ou altera os objetos virtuais [Kirner e Siscouto, 2007].

Segundo Frery et al. (2002), o paradigma 3D é útil porque oferece a possibilidade de representar a informação de um modo realísta, organizando o conteúdo de uma maneira espacial. Isto possibilita um aumento na percepção das informações contidas no ambiente e permite ao usuário explorar a informação de um modo interativo e dinâmico.

As pesquisas realizadas com AV têm como objetivo reduzir os problemas de acessibilidade e usabilidade de ambientes interativos 3D. Tais pesquisas apresentam abordagens diferentes para algumas questões, como por exemplo: quais modificações devem ser realizadas e em que momento; quais informações sobre o usuário são necessárias; e quais formas de adaptação de conteúdo existem.

Técnicas de adaptação têm sido desenvolvidas para personalizarem os AV, construídos de acordo com as preferências de cada usuário, seu nível de conhecimento e suas necessidades. Algumas dessas técnicas estão baseadas na ajuda ao usuário durante a navegação [Rickel e Johnson, 2000; Rikel et al., 2002; Chittaro et al., 2003] e na

definição de prioridades para apresentação de conteúdos (os produtos mais acessados ou consultados em comércio eletrônico, por exemplo) [Wernet e Hanson, 1999; Walsack e Cellary, 2002].

A adaptação durante a navegação pode ser realizada utilizando algumas técnicas de ajuda ao usuário ou de acompanhamento das suas ações. De acordo com Ballegooij e Eliëns (2001), diversos problemas de exploração em AV são relatados pelos usuários, tais como identificação da estrutura da cena, desorientação e dificuldade de navegação, que muitas vezes dependem do conhecimento adquirido em uma atividade anterior. Neste caso, a assistência ao usuário se faz necessária tanto para a navegação pelo ambiente, como para ajudar na localização de informações. Diferentes soluções têm sido propostas para esses problemas: acompanhamento (navegação guiada) de uma estrutura de cena [Pitarello, 2001; Pitarello e Celentano, 2001]; inserção de marcadores (*landmarks*) [Vinson, 1999]; e assistentes virtuais de navegação.

A introdução de assistentes virtuais tem a vantagem adicional de orientar o usuário e tornar o ambiente mais atrativo para ele. Assim, o uso de assistentes virtuais pode enriquecer a interação com o ambiente e evitar que os usuários se sintam perdidos no ambiente [Chittaro e Ranon, 2002a].

A adaptação para apresentação de conteúdos em função das características do usuário é, cada vez mais, considerada um dos fatores preponderantes na satisfação dos usuários do sistema. Em um ambiente 3D, a organização espacial dos seus objetos pode facilitar a navegação e exploração, contribuindo para uma interação mais efetiva do usuário.

Segundo Santos (2004), esta organização exige certos agrupamentos de acordo com algum critério semântico. Por exemplo, em uma loja virtual, é interessante o agrupamento dos produtos conforme a seção a que pertencem (eletrônicos, bazar, vestuário, entre outros) ou, em um ambiente de apoio à Educação a Distância, é necessário agrupar espacialmente os conteúdos conforme as áreas de conhecimento a que pertencem.

Entretanto, apesar dos diversos trabalhos realizados sobre adaptação de AV, muito pouco tem sido discutido sobre a adaptação do ambiente em função da apresentação de conteúdos com vários níveis de informação. Aplicações desse tipo podem apresentar um ambiente com uma riqueza maior ou menor de detalhes em

função do nível de conhecimento do usuário ou de suas necessidades. Por exemplo, um Ambiente Virtual pode apresentar uma complexidade baixa de detalhes para usuários inexperientes ou que não estão familiarizados com o ambiente visitado. Caso contrário, para os mais experientes, ou que aprenderam durante a interação, a complexidade de informações do mundo pode ser aumentada, gerando um ambiente mais completo e mais interessante para o aluno.

Esse tipo de personalização pode ser útil para ambientes virtuais voltados para o ensino a distância, em que diversos alunos com conhecimentos distintos podem se conectar remotamente e acessar um AV específico para cada perfil; ou então, em ambientes de simulação que permitem gerar níveis de complexidade para uma mesma situação em função do perfil do usuário. Também na área de jogos, a personalização pode ser utilizada para gerar AV com níveis de dificuldades diferentes.

1.1 Motivação

A evolução das tecnologias de Realidade Virtual (RV), Inteligência Artificial e de comunicação na Web, vêm tornando possível a construção de AV mais voltados para o usuário, incorporando procedimentos de acompanhamento das suas ações e das modificações do ambiente. Neste novo paradigma, elementos dinâmicos e interativos são introduzidos de forma a aumentar a participação do usuário durante a navegação no ambiente. Esta participação está diretamente relacionada com o ambiente utilizado e com o nível de retorno que o sistema propicia. Pesquisas nessa área têm desenvolvido técnicas para gerarem AV que se adaptam em função do perfil do usuário, modelando o mundo de acordo com as características de cada perfil.

Segundo Chittaro e Ranon (2002a), a capacidade de adaptar o conteúdo, a estrutura e a apresentação do ambiente em função das características do usuário, é cada vez mais considerada um fator chave para incrementar o seu nível de satisfação. Em um shopping virtual, por exemplo, a possibilidade de alterar a disposição das lojas conforme os interesses do usuário (impossível de ser feita em um shopping real, e pouco efetiva em um ambiente bidimensional) pode tornar a interface mais amigável e atrativa.

Existem basicamente três problemas identificados durante as pesquisas bibliográficas realizadas neste trabalho que ainda não foram completamente solucionados. O primeiro refere-se à inexistência de uma adaptação no detalhamento do

conteúdo a ser apresentado ao usuário. Ambientes virtuais que se adaptam em função das características do usuário são descritos, por exemplo, em Chittaro e Ranon (2002a) e Santos (2004). Esses trabalhos propõem uma estruturação mais adequada para a organização espacial das informações, de acordo com os interesses do usuário. Entretanto, em ambos os casos, as informações fornecidas na construção do ambiente são sempre as mesmas; o que ocorre é apenas um rearranjo dos objetos do mundo para que se adapte aos interesses do usuário ou para permitir o acesso às informações que o usuário está precisando.

O segundo problema em aberto é a adaptatividade de AV em tempo real, ou seja, a capacidade do sistema em poder modificar as informações do mundo virtual durante uma mesma sessão. Nos trabalhos referenciados acima, somente nas próximas interações do usuário com o ambiente é que as modificações são realizadas.

Finalmente, o último problema é a construção de mundos virtuais utilizando objetos 3D armazenados em um Sistema de Gerenciamento de Banco de Dados (SGBD). As pesquisas referenciadas na literatura não tratam do problema. A maioria delas, por utilizarem arquivos VRML [Ames et al., 1997; Web3D, 2007b], armazena os objetos do mundo em um único arquivo, não realizando um gerenciamento de tais objetos e, conseqüentemente, não explorando a possibilidade de reuso de objetos. O armazenamento e a recuperação de objetos em um SGBD podem ser realizados por linguagens de consulta com suporte a XML [XML, 2007], obedecendo à especificação do mundo a ser criado.

Para poder resolver os problemas em aberto descritos acima, foi feito um levantamento das principais limitações encontradas em aplicações que utilizam AV adaptados ao perfil do usuário e as soluções que motivaram a realização deste trabalho:

- O desenvolvimento de AV interativos e imersivos tem custo alto e requer equipamentos especiais. É necessário encontrar soluções mais eficientes e menos onerosas;
- A interação do usuário está diretamente relacionada com a motivação. Os ambientes precisam se adaptar aos interesses e à evolução cognitiva do usuário. É necessário considerar o conhecimento prévio do usuário sobre os AV a serem visitados, suas preferências de uso e de navegação, bem como a sua evolução durante a interação com o AV;

- Os ambientes virtuais voltados para Web requerem tecnologias eficientes de comunicação, acesso às informações, geração e monitoração do ambiente, entre outros. O surgimento de novas tecnologias (baseadas em XML, principalmente) facilita o desenvolvimento de tais ambientes;
- Os usuários necessitam ser monitorados para que se possa identificar seu perfil e as suas ações durante a interação com o ambiente. Este processo permite conhecer as preferências do usuário, bem como, sua capacidade cognitiva e seu conhecimento sobre o ambiente. O uso de agentes inteligentes tem apresentado resultados bastante satisfatórios, tendo em vista que tais agentes possuem sensores para reconhecer o comportamento do usuário e uma base de conhecimento que os auxilia no processo de identificação do seu perfil;
- AV personalizados realizam intenso processo de inferência. Métodos de inferência tradicionais (que concentram processamento em uma única base de conhecimento) tornam-se lentos e ineficientes. A aplicação de agentes inteligentes e conseqüente distribuição de tarefas têm propiciado grandes avanços nessa área [Eiter e Mascardi, 2002; De Antonio, 2005]
- Os mundos virtuais são via de regra armazenados em arquivos VRML e, portanto, eles só podem ser visualizados como um todo, não podendo ser utilizado apenas uma parte deles. O uso de novas técnicas de representação de objetos 3D (X3D [X3D, 2007a], por exemplo) e de gerenciamento desses objetos em um SGBD XML permitem reutilizar os componentes do ambiente em novos mundos, reduzindo o espaço de armazenamento desses mundos e aumentando a eficiência e eficácia na geração de novos AV;
- A adaptação de ambientes ainda é realizada *off-line*. Mesmo que o usuário adquira conhecimento sobre o conteúdo do mundo, a adaptação só ocorre na sessão seguinte. Adaptação em tempo real ainda é um problema em aberto, mas que pode ser resolvido parcialmente com as novas técnicas de armazenamento, recuperação e transporte de objetos XML; e
- A maioria dos AV não guarda o estado atual do ambiente e as modificações realizadas pelo usuário. É importante que na próxima interação do usuário, o ambiente atualizado seja recuperado. Para tanto, é necessário realizar um gerenciamento do estado atual do AV e armazená-lo adequadamente em um SGBD.

1.2 Objetivos

Esta tese tem como objetivo propor, construir e validar uma infra-estrutura para gerenciamento de componentes de ambientes virtuais adaptativos, em tempo real, construídos a partir de bases de objetos reutilizáveis.

A infra-estrutura desenvolvida, denominada VEPersonal (*Personalized Virtual Environment*), utiliza um modelo de representação de objetos 3D armazenados em um SGBD com suporte a XML, no qual cada objeto contém vários níveis de informação.

O VEPersonal deve monitorar as ações do usuário e realizar a adaptação do mundo virtual de acordo com o perfil do referido usuário. Esta adaptação pode ocorrer tanto em função das preferências e interesses do usuário, definidas no início de uma sessão, como também em função do conhecimento adquirido durante a interação com o sistema.

Esta infra-estrutura é controlada por uma sociedade de agentes responsável por identificar e armazenar o perfil do usuário e as modificações ocorridas no ambiente, bem como determinar as adaptações necessárias ao AV.

1.2.1 Objetivos Específicos

O VEPersonal tem o objetivo de solucionar os problemas de adaptatividade descritos neste capítulo e, para tanto, deve atender aos seguintes objetivos específicos:

- Gerar e manter ambientes virtuais utilizando ferramentas com suporte a Web baseadas em tecnologias que utilizem XML;
- Gerenciar e recuperar objetos 3D para facilitar o reuso na construção de novos mundos;
- Gerar conteúdos com vários níveis de informação e utilizá-los de acordo com as características do perfil do usuário; e
- Gerar ambientes personalizados em tempo real, atualizando o AV em função do conhecimento adquirido pelo usuário durante a sua interação.

1.3 Organização do Trabalho

Este trabalho utiliza basicamente quatro áreas de pesquisa para a construção da infra-estrutura proposta: Ambientes Virtuais, Sistemas de Gerenciamento de Banco de Dados, Agentes e Interface. A Figura 1.1 apresenta uma visão esquemática dos recursos utilizados e sua inserção na tese ora apresentada.

Figura 1.1 Áreas de pesquisa utilizadas na construção do VEPersonal

A geração de ambientes virtuais e o uso de linguagens de representação de objetos 3D permitem a construção de AV dinâmicos e interativos. A adaptatividade e a modelagem do usuário foram incluídas nesta área para darem subsídios ao desenvolvimento do AV adaptativo. Um SGBD com suporte a XML é utilizado para o armazenamento e gerenciamento de objetos 3D. Uma sociedade de agentes permite monitorar as ações do usuário e propor as modificações durante a atualização do AV. Por fim, a construção de uma interface entre o sistema gerenciador da aplicação e a máquina do usuário se faz necessária, pois integra os recursos desenvolvidos entre as ferramentas utilizadas.

Esta tese está estruturada em seis capítulos. Além deste, os demais capítulos foram organizados como segue.

O capítulo 2 discute a importância da geração de ambientes virtuais e as vantagens da realidade virtual na melhoria da interação dos usuários nesses ambientes. É abordado também o uso da Inteligência Artificial em ambientes tridimensionais e sua contribuição para o aprendizado do usuário. Uma avaliação dos ambientes descritos nesse capítulo é realizada, procurando discutir a sua importância para o desenvolvimento do trabalho apresentado nesta tese.

O capítulo 3 apresenta uma avaliação das tecnologias existentes para a construção de um AV com o objetivo de identificar os recursos necessários para o desenvolvimento da infra-estrutura proposta. Inicialmente, este capítulo aborda o uso de um SGBD para o gerenciamento de objetos tridimensionais, realizando uma comparação entre os SGBD nativos e os SGBD relacionais com suporte a XML. Em seguida, ainda neste capítulo, é feita uma avaliação das técnicas de Inteligência Artificial para modelagem do usuário, descrevendo o Modelo do Usuário que irá armazenar o seu perfil. Um estudo sobre agentes inteligentes também é realizado neste capítulo, procurando discutir a importância do uso de uma sociedade de agentes em AV adaptativos.

No capítulo 4 é apresentada a arquitetura do VEPersonal, descrevendo a estrutura utilizada e as soluções encontradas para as tecnologias discutidas no capítulo anterior que ajudaram na construção do sistema. Também neste capítulo é apresentada uma proposta de visualização de níveis de complexidade de informação para cada objeto. Esta proposta representa uma solução ao problema de adaptatividade em tempo real em função do nível de conhecimento que o usuário tem sobre o mundo e de sua capacidade cognitiva.

A implementação da arquitetura, bem como os resultados alcançados e uma avaliação do desempenho do VEPersonal são apresentados no capítulo 5. No capítulo 6 são apresentadas as conclusões obtidas nesta tese e a sugestão de trabalhos futuros.

2. Ambientes Virtuais

Um ambiente virtual pode simular um ambiente real ou construir ambientes imaginários, permitindo a um ou mais usuários interagirem através da visualização e manipulação de representações extremamente complexas. Para isso, essa interação é realizada com o uso de técnicas e de equipamentos computacionais que ajudam na ampliação do sentimento de presença do usuário no ambiente [Teichrieb, 1999].

O advento da Realidade Virtual e o avanço de recursos computacionais possibilitaram a utilização de objetos em três dimensões, além de vários recursos multimídia como imagens, vídeo e som, tornando os ambientes virtuais mais realistas e ao mesmo tempo mais interativos. Segundo Tori e Kirner (2006, p.3):

> *"... os sentidos e as capacidades das pessoas podem ser ampliados em intensidade, no tempo e no espaço. É possível ver, ouvir, sentir, acionar e viajar muito além das capacidades humanas ... Pode-se, assim, ser tão grande (em nível das galáxias) ou tão pequeno (em nível das estruturas atômicas) quanto se queira, viajando a velocidades muito superiores a da luz e aplicando forças descomunais. Ao mesmo tempo, pode-se ampliar a medida do tempo, para que as pessoas possam observar ocorrências muito rápidas em frações de segundos, implementando o conceito de câmera lenta, ou reduzir a medida do tempo, acelerando-o, para observar ocorrências e fenômenos muito lentos, que poderiam demorar séculos. Para isto, são utilizadas técnicas de modelagem tridimensional na elaboração dos objetos e montagem do cenário virtual, por onde o usuário poderá navegar".*

Do ponto de vista da interação do usuário com o mundo virtual, os ambientes virtuais podem apresentar duas categorias: imersivo e não-imersivo. Os AV imersivos são baseados no uso de dispositivos sofisticados (capacetes, luvas, cavernas e seus acessórios, por exemplo) que capturam os movimentos e comportamentos do usuário e reagem a eles. Estes sistemas são utilizados para exibição de aplicações de RV baseadas em projeção, nos quais o usuário é transportado predominantemente para o domínio da aplicação. Os AV não-imersivos baseiam-se no uso de equipamentos *desktop*, *notebook*

e outros dispositivos com display 2D. Neste caso, o usuário é transportado apenas parcialmente ao mundo virtual.

Embora os AV imersivos tenham evoluído e apresentem aplicações muito realistas, os sistemas não-imersivos são os mais populares por serem mais baratos e mais simples de implementar. Além disso, novas tecnologias proporcionam uma visualização de melhor qualidade para os ambientes não-imersivos, aumentando assim a sensação tridimensional. Como ponto negativo dessa forma de interação está a falta de sensação de imersão no ambiente por parte do usuário.

Outra forma de interação com AV é a inserção de objetos virtuais no ambiente físico, mostrada ao usuário, em tempo real, com o apoio de algum dispositivo tecnológico. Ambientes com tais características são denominados ambientes de Realidade Aumentada [Kirner et al., 2004]. Nesses ambientes, o usuário permanece no seu ambiente real e interage com os objetos do mundo virtual de maneira natural e realista.

Interfaces multimodais têm sido desenvolvidas para que o usuário possa manipular objetos virtuais usando as mãos ou outros dispositivos de simples interação. O objetivo é eliminar o inconveniente dos aparatos tecnológicos ainda existentes na reprodução de mundos para a Realidade Aumentada [Kirner e Tori, 2006]. Ainda, segundo Kirner e Tori (2006, p.23):

> *"...a realidade virtual e a realidade aumentada permitem ao usuário retratar e interagir com situações imaginárias, como os cenários de ficção, envolvendo objetos reais e virtuais estáticos e em movimento. Permitem também reproduzir, com fidelidade, ambientes da vida real como a casa virtual, a universidade virtual, o banco virtual, a cidade virtual, etc., de forma que o usuário possa entrar nesses ambientes e interagir com seus recursos de forma natural, usando as mãos (com ou sem aparatos tecnológicos, como a luva) e eventualmente comandos de voz. Com isto, o usuário pode visitar salas de aula e laboratórios de universidades virtuais, interagindo com professores e colegas e realizando experimentos científicos; pode entrar no banco virtual e manusear o terminal de atendimento virtual, da mesma maneira que o faz com o equipamento real, e mesmo conversar com o gerente, representado no ambiente por um humanóide virtual".*

Como visto, diversos fatores têm motivado a construção de ambientes virtuais. Mais recentemente, a integração de técnicas de Inteligência Artificial aos AV tem permitido a representação de ambientes que exploram o uso de entidades com inteligência e diferentes formas de interações, provendo maior dinamicidade, realismo e usabilidade aos ambientes [Santos, 2004]. Tais ambientes são conhecidos como Ambientes Virtuais Inteligentes (AVI).

Na seção 2.1, são analisados alguns exemplos de AVI e discutida a sua importância para a geração de ambientes 3D, realistas e interativos. Os critérios utilizados para avaliação desses AV foram: tipo de representação do ambiente, utilizando objetos 3D dinâmicos e interativos; uso de agentes inteligentes em AV para o acompanhamento das ações do usuário; capacidade dos AV em identificar o perfil do usuário e determinar o Modelo do Usuário; capacidade dos AV em modificar os componentes do mundo virtual em função do perfil do usuário; e tecnologia utilizada.

Considerando a necessidade de gerar ambientes que se adaptam ao perfil do usuário, a seção 2.2 apresenta alguns exemplos de AV Inteligentes, cuja principal característica é a capacidade de adaptação desses ambientes ao usuário. Ambientes com essa característica são denominados nesta tese de AV Adaptativos.

Na seção 2.3, é realizada uma análise a respeito das deficiências (pontos fracos) e vantagens (pontos fortes) encontradas nos ambientes estudados. Particularmente, esta análise procura identificar as limitações ainda existentes na área, de modo a tratá-las no desenvolvimento da personalização de ambientes virtuais, fruto deste trabalho de tese.

2.1 Ambientes Virtuais Inteligentes

Segundo Anastassakis et al. (2001), um AVI pode ser definido como um Ambiente Virtual semelhante ao mundo real, habitado por entidades autônomas inteligentes e exibindo uma variedade de comportamentos. Essas entidades podem ser objetos estáticos simples ou dinâmicos, representações virtuais de formas de vida (humanos e animais) ou *avatares* (representação do usuário no ambiente).

De acordo com Rickel et al. (2002), Gratch et al. (2002) e Anastassakis et al. (2001), aplicações de AVI têm sido empregadas em diversas áreas, principalmente para simulação, entretenimento e educação. Em simulação, por exemplo, diferentes tipos de

ambientes habitados por humanos virtuais podem auxiliar em projetos arquitetônicos ou no controle de tráfego. Também tem sido explorada a simulação de humanos virtuais em situações de emergência [Musse, 2000] e de comportamentos de grupos de animais [Reynolds, 2001].

Na área de entretenimento, alguns exemplos de aplicações são os jogos com cenários que podem ser modificados conforme o andamento do jogo [Grand e Cliff, 1998] ou ambientes onde o usuário pode navegar e interagir com outros usuários e com assistentes virtuais, como por exemplo, teatros [Nijholt e Hulstijn, 2000], museus [Devillers et al., 2002] e lojas virtuais [Chittaro e Ranon, 2002a]. Aplicações com estórias interativas [Cavazza et al., 2001], em que o usuário é um participante ativo e pode interferir no curso das mesmas, têm surgido como uma nova forma de entretenimento.

Na área educacional, a incorporação de personagens tutores [Rickel e Johnson, 1997] e a exploração de interações multi-modais, juntamente com sofisticadas técnicas de representações da informação, podem prover experiências de aprendizado cada vez mais agradáveis e efetivas.

Nesta seção, os AVI analisados possuem as seguintes características: um ambiente é considerado "dinâmico" caso possua objetos que se movimentem ou que apresentem a possibilidade de realizar movimentos quando estimulados. Ambiente "estático" é para os casos em que não existam essas possibilidades. Além disso, um ambiente pode possuir apenas um "único" agente, ou "vários" caso existam múltiplos agentes interagindo no ambiente. No caso de um ambiente poder modificar seu conteúdo em função das características do usuário, ele é cosiderado "adaptável". Caso contrário, ele é considerado "não adaptável".

Um ambiente virtual para ensinar procedimentos de operações navais foi desenvolvido por Johnson (2001). Neste ambiente, existe um agente animado, denominado STEVE (*Soar Training Expert for Virtual Environments*), que ensina aos usuários o funcionamento de motores e equipamentos (Figura 2.1). O agente STEVE além de demonstrar as tarefas a serem realizadas e explicar suas ações, monitora a performance das tarefas dos estudantes, auxiliando-os quando necessário.

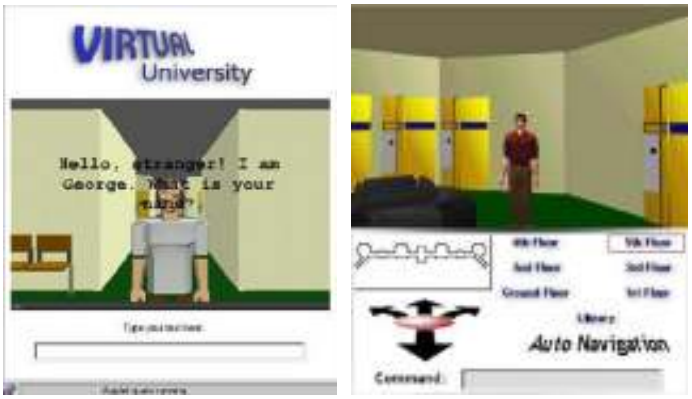
É um ambiente de treinamento, utilizando um agente para interação com o usuário. Este sistema não armazena o perfil do usuário e consequentemente não

considera o nível de conhecimento do aluno nem o grau de dificuldade do assunto. Neste caso, é o aluno que tem que se adaptar ao conteúdo apresentado.

Figura 2.1 Steve descrevendo o funcionamento de um painel [Johnson et al., 1998]

Panayiotopoulos et al. (1999) apresentam uma Universidade Virtual, em que um guia virtual conduz os visitantes a lugares relevantes, de acordo com as necessidades de informação dos mesmos, além de apresentar documentos multimídia apropriados. O agente virtual se comunica com os usuários, os quais podem indicar suas necessidades de informação através de linhas de comando. As Figuras 2.2 (a) e (b) apresentam a interface de entrada ao ambiente e a movimentação do agente.

É importante o auxilio à navegação neste ambiente, através do uso de um agente. O ambiente é estático e não há adaptação de conteúdo de acordo com as necessidades e/ou preferências do usuário, não sendo realizada uma avaliação do perfil.



(a) (b)

Figura 2.2 Visita à Universidade Virtual acompanhada de um agente virtual. (a) Interface de entrada ao ambiente [Panayiotopoulos et al., 1999]; (b) Agente se deslocando no ambiente [Vosinakis e Panayiotopoulos, 2005]

Nijholt e Hulstijn (2000) apresentam um teatro virtual, em que os visitantes navegam e interagem com agentes que possuem informações sobre shows, músicos e

ingressos. Os usuários também podem visitar salas de concerto e admirar quadros, sendo ajudados pelos agentes durante a navegação. A Figura 2.3 apresenta o agente virtual interagindo com o visitante.

Da mesma forma que o AVI anterior, este ambiente não é dinâmico, nem é adaptável, como também não armazena o perfil do usuário.

Figura 2.3 Teatro Virtual: Agente Virtual interagindo com o visitante [Nijholt e Hulstijn, 2000]

Anastassakis et al. (2001) apresentam uma Biblioteca Virtual, construída em VRML, onde dois agentes (cliente e bibliotecário) se comunicam: o agente cliente requisita informações sobre livros ao agente bibliotecário, responsável por localizar e apresentar o item desejado. A Figura 2.4 apresenta a interface do ambiente.

Figura 2.4 Interação entre os agentes cliente e bibliotecário [Anastassakis et al., 2001]

O agente bibliotecário já possui características mais evoluídas, procurando auxiliar o usuário na sua pesquisa e propondo, através de diálogo em linguagem natural, soluções alternativas caso não encontre o item desejado.

O agente, por não conhecer o perfil do usuário, tem que realizar um diálogo para sugerir outras opções. O ambiente é estático, não armazena o perfil do usuário e não se adapta às necessidades do usuário.

Em Rizzo et al. (2002) e Rizzo et al. (2006) são apresentadas salas virtuais tridimensionais em que um avatar representa um professor e objetos se movimentam pelo ambiente. A interação de crianças com o ambiente é monitorada e os dados coletados são usados para análise do comportamento das mesmas, no tratamento de crianças hiperativas. A Figura 2.5 ilustra um exemplo de uma sala de aula virtual.

O ambiente é dinâmico, gerando maior motivação e interação por parte dos usuários. Ele não armazena, nem considera o perfil do usuário. A análise do comportamento é uma característica importante que poderia ser utilizada para adaptação do ambiente, embora neste caso, não ocorra nenhum tipo de adaptação.

Figura 2.5 Sala de aula virtual [Rizzo et al., 2006]

O Active Worlds [ActiveWorlds, 2007] consiste em um conjunto de ambientes 3D, disponível comercialmente, onde o usuário, representado por um avatar, pode navegar e interagir com outros usuários. Agentes virtuais, que se deslocam no ambiente de forma pré-programada, são utilizados para popular os ambientes, mas não interagem com o usuário para troca de informações. A Figura 2.6 (a) e (b) apresentam exemplos de interfaces do ambiente do Active Worlds.

(a) (b)

Figura 2.6 Interfaces de ambientes do Active Worlds. (a) Visitando uma cidade histórica (b) Visitando um shoping [ActiveWorlds, 2007]

Estas interfaces utilizam recursos de animação para gerar um ambiente dinâmico. Entretanto, este ambiente não modifica as suas características em função da evolução da interação do usuário e também não armazena o perfil do usuário.

Dos ambientes virtuais existentes atualmente, o Second Life [SecondLife, 2007a] surge com um novo conceito de representação de AV. Utilizando o conceito de 'ilhas' no lugar de sites, de 'trilhas' substituindo os menus, de navegação sem a necessidade de utilização de um browser e de código aberto para a construção de objetos, o Second Life tornou-se um ambiente colaborativo tridimensional de realidade virtual completamente criado por seus residentes (pessoas que acessam o sistema, através de um *avatar*, e habitam o local) [Mattar e Maia, 2007]. Neste ambiente, vários usuários podem visitar as ilhas, conversar com os outros visitantes, correr, trabalhar, construir, comprar e vender objetos, socializar e brincar.

Desenvolvido em 2003 pela empresa Linden Research [Linden, 2007], ele implementa praticamente todas as características presentes num mundo real e outras só possíveis num ambiente de RV tais como: voar e tele-transportar. As Figuras 2.7 (a), (b) e (c) apresentam exemplos de interfaces do ambiente Second Life.

(a) (b) (c)

Figura 2.7 Interfaces de ambientes do Second Life. (a) Visitando uma ilha de informações jornalísticas (b) Sobrevoando ilhas temáticas (c) Interagindo em ilha de relacionamento [SecondLife, 2007b]

Navegar nesse ambiente pode ser entendido como uma mistura de realidade virtual com site de relacionamentos. Usuários conectam-se e se comunicam uns com os outros, o que torna possível a realização de atividades em grupo. A construção de mundos virtuais, também conhecidos como Metaverso, pode ser realizada pelos residentes, utilizando a linguagem de *scripts* Linden Scripting Language [LSL, 2007].

O *Second Life* possui sua própria moeda, o *Linden dollar* (L$), que pode ser convertida em dólares verdadeiros. Há inúmeras maneiras de se obter uma fonte de

renda: criar objetos, construir imóveis, desenvolver acessórios para os *avatares,* entre outros. Entretanto, ganhar dinheiro no *Second Life* não é fácil. Por haver atividade econômica e mercado, as barreiras e oportunidades são as mesmas que na vida real.

Por outro lado, muitas universidades estão usando o *Second Life* como veículo de ensino e aprendizagem. As instituições educacionais que participam desse ambiente ainda têm atuação limitada e, na sua maioria, estão presentes apenas para ocupar espaço no Metaverso e promover o marketing da instituição [Mattar e Maia, 2007].

Contudo, propostas inovadoras surgem nas aplicações do *Second Life*. As empresas e universidades estão substituindo a vontade unilateral de apenas lucrar, pela coletiva de gerar conteúdos gratuitos, explorar recursos e criar uma rede colaborativa de avatares. Por exemplo, "A Cidade do Conhecimento 2.0", incubadora virtual ligada à Universidade de São Paulo – USP; o Centro Cultural Bradesco, ilha para promover debates sobre arte, filosofia e tecnologia; e ilhas inteiras dedicadas à educação gratuita, como a que ensina o mapeamento genético humano e explicações intuitivas sobre adenina e citosina. [Estadao, 2007]. Tais experiências viram crescer em volta de si uma rede de freqüentadores que não usam o *Second Life* como escapismo nem gastam horas e horas on-line. Isto pode tornar-se uma nova forma de comunicação entre as pessoas.

No Second Life, o ambiente é dinâmico, abrindo enormes possibilidades de interação por parte dos usuários. As ilhas voltadas para o ensino ainda não consideram o perfil do usuário como um fator de adaptação ou direcionamento do seu conteúdo, mas o uso de ambientes colaborativos nessas ilhas já as tornam com um grande potencial para aplicações voltadas para os interesses e preferências do usuário.

No Quadro 2.1, é apresentada uma comparação entre os Ambientes Virtuais Inteligentes abordados nesta seção. Os agentes foram utilizados nesses ambientes para auxiliar o usuário durante sua interação com o ambiente. Nos AV estudados, o auxílio é realizado mediante o esclarecimento de questões colocadas pelo usuário, ou na busca de informações e de locais por ele solicitados.

Considerando os exemplos apresentados, verifica-se que esses ambientes não tratam o conhecimento prévio do usuário e nem o acompanhamento de sua evolução durante a interação com o ambiente. As técnicas de IA utilizadas não são direcionadas para identificar o perfil do usuário, nem tão pouco para adaptar o ambiente virtual às necessidades do mesmo.

Quadro 2.1 Comparativo entre os AV Inteligentes

Requisito / Sistema	Tipo de Ambiente	Uso de Agente	Interação Agente/ Usuário	Perfil Usuário	Adaptável (Ambiente)	Tecnologia utilizada
Johnson et al. (2001)	Treinamento, 3D, dinâmico	Representação gráfica, único	Indicação de tarefas a realizar no ambiente	Não	Não	VRML
Panayiotopoulos et al. (1999 e 2005)	Entretenimento, 3D/2D, estático	Representação gráfica, único	Consulta em LN, auxílio a navegação	Não	Não	VRML
Nijholt e Hulstijn (2000)	Entretenimento, 3D, estático	Repres. gráfica, conhec. do ambiente, único	Diálogo em LN, auxílio navegação	Não	Não	VRML
Anastassakis et al. (2001)	Genérico, 3D/2D, estático	Repres. gráfica, reativo, cognitivo, vários	Consulta em LN	Não	Não	VRML, OpenGL
Rizzo et al. (2002 e 2006)	Diagnóstico cognitivo, 3D, modificável, dinâmico	Repres. gráfica, indicação de tarefas a realizar, vários	Indicação de tarefas a realizar no ambiente	Não	Não	(*)
Active Worlds (2007)	Genérico, 3D, dinâmico	Representação gráfica, vários	Não tem (apenas movimentação dos avatares)	Não	Não	DirectX
Second Life (2007)	Genérico, 3D, dinâmico	Representação gráfica, vários	Não tem (apenas movimentação dos avatares)	Não	Não	Linden Scripting Language

[*] Não informado pelo autor

2.2 Ambientes Virtuais Adaptativos

Segundo Celentano et al. (2004), um Ambiente Virtual adaptativo deve possuir inteligência para atualizar o ambiente 3D com o objetivo de reduzir a necessidade cognitiva de interação. Desta forma, o sistema pode auxiliar o usuário durante a interação, alterando a complexidade do ambiente ou facilitando no processo de navegação.

Um AV Adaptativo deve identificar o perfil do usuário no início de uma sessão e determinar o ambiente mais adequado a ser gerado. Para isso, a representação dos mundos com os quais este ambiente vai trabalhar tem que permitir a sua manipulação através da inserção e remoção de objetos e atualizações de forma eficiente.

Neste contexto, a evolução dos AV e a necessidade de gerar aplicações direcionadas às preferências e interesses do usuário exigiram a inclusão de novas técnicas para construir AV mais próximos da realidade. As técnicas de modelagem do usuário, por exemplo, foram incorporadas a essas aplicações com o objetivo de identificar as características do usuário, suas necessidades e preferências, além de avaliar o conhecimento do mesmo sobre o ambiente.

Agentes inteligentes são muito úteis em aplicações de AV adaptativos, uma vez que eles podem monitorar as ações do usuário através de sensores, identificando as interações realizadas com os objetos do mundo e, através de atuadores, realizar modificações no ambiente virtual.

Nesta seção, são apresentados exemplos de AV Adaptativos e as técnicas de adaptação utilizadas. Estas técnicas são empregadas tanto para a organização dinâmica dos objetos no ambiente (por ex., produtos mais procurados pelo cliente são mostrados em primeiro plano em uma loja virtual), facilitando a navegação, a exploração e uma interação mais efetiva; como também para a adaptação do nível de complexidade do objeto ao grau de informação que o usuário tem sobre aquele ambiente, contribuindo para a adequação do AV em função das características e necessidades do usuário.

A utilização de avatares como guias interativos em ambientes tridimensionais é proposta por Frery et al. (2002). Com base no conhecimento sobre o usuário, coletado a partir de formulários (coleta explícita), o perfil do usuário é identificado e armazenado no Modelo do Usuário. Com essas informações, a estratégia de navegação pelo

ambiente é definida, bem como a representação gráfica do avatar que irá representar o usuário.

A Figura 2.8 apresenta a arquitetura do avatar em que se pode identificar a existência de um módulo para armazenar o perfil do usuário e um motor de inferência responsável por determinar o comportamento do avatar.

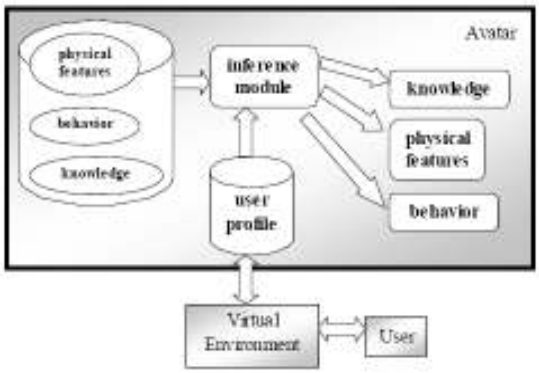


Figura 2.8 Arquitetura do Avatar [Frery et al., 2002]

Neste caso, o ambiente é adaptável e dinâmico. Apesar de realizar apenas uma coleta explícita no início de uma sessão, já existe a preocupação com a identificação das características do usuário e sua utilização para construção de ambientes adaptativos. Nessa proposta, tais características são utilizadas para gerar roteiros de apresentação de conteúdos.

Chittaro e Ranon (2002a) desenvolveram o AWE3D (Adaptive WEb 3D), um AV adaptativo que representa uma loja virtual em um ambiente Web, no qual os usuários podem navegar e obter informações sobre os objetos espalhados pelo ambiente. As informações sobre os interesses do usuário, utilizadas para a personalização do ambiente, são realizadas através da coleta explícita e da monitoração das ações do usuário no ambiente (tais como produtos visualizados e compras efetuadas). As Figuras 2.9 (a) e (b) apresentam as adaptações no ambiente e os objetos que trafegam pelo mesmo.

Figura 2.9 Navegação em uma Loja Virtual. (a) Produtos em promoção aparecem em destaque, (b) Produtos organizados em função do perfil de compra do usuário. [Chittaro e Ranon, 2002a]

A arquitetura do AWE3D (Figura 2.10) possui uma base de conhecimentos (BC) responsável pela atualização do modelo do usuário e pela personalização do ambiente Web. É através das regras existentes na BC que novos mundos virtuais personalizados são criados.

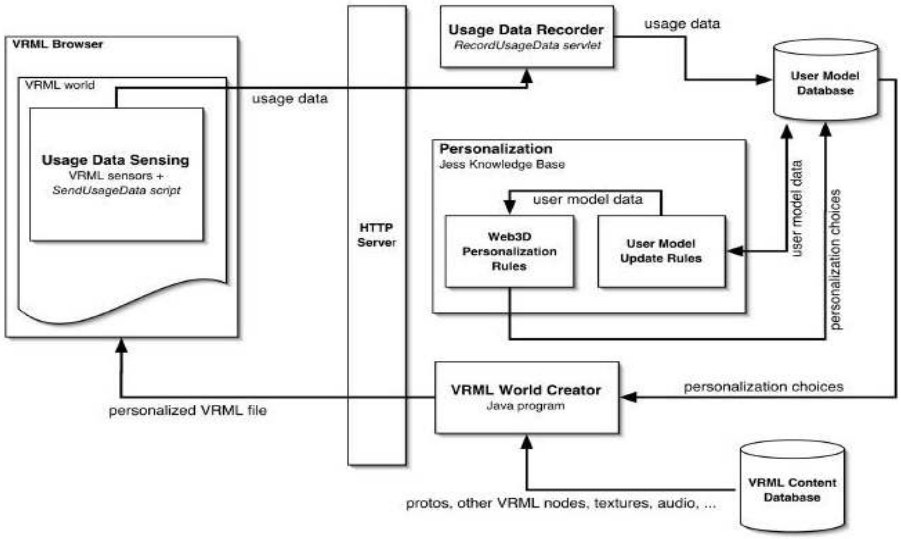
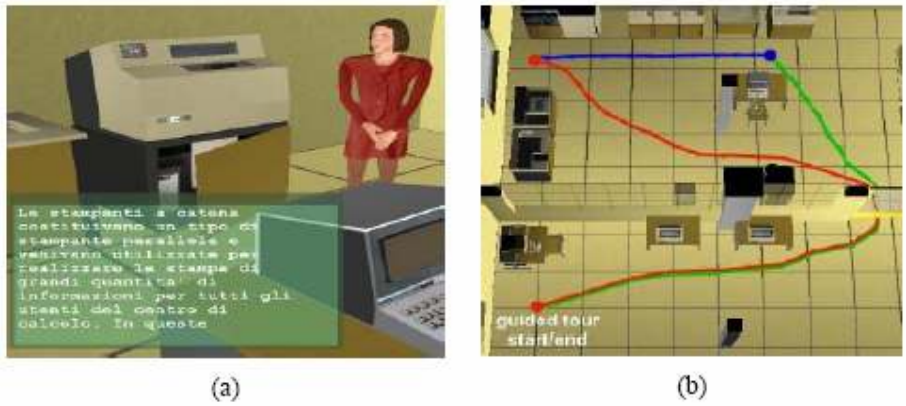


Figura 2.10 Arquitetura do AWE3D para adaptação em ambiente Web [Chittaro e Ranon, 2002a]

Este ambiente é dinâmico (possuindo maior interatividade com o usuário) e adaptável, pois a apresentação de objetos varia em função do perfil de compra do cliente, o qual considera suas necessidades, preferências, visualização de produtos e opções de compra.

Em um trabalho posterior, Chittaro et al. (2003) apresentam um agente virtual designado a auxiliar o usuário na navegação por um museu virtual. A partir da descrição dos lugares ou objetos de interesse a serem visitados no museu, fornecidos no início de uma sessão, o agente cria uma trajetória apropriada. Além disso, o agente está apto a parar durante o trajeto e apresentar cada objeto ou lugar. As Figuras 2.11 (a) e (b) apresentam o agente em uma explanação sobre um dos objetos do ambiente e uma trajetória criada, respectivamente.



(a) (b)

Figura 2.11 (a) Agente explanando sobre o objeto; (b) Trajetória elaborada para a navegação pelo ambiente [Chittaro et al., 2003]

A trajetória é calculada pelo Gerenciador de Trajetos, que tem como função derivar uma seqüência apropriada dos pontos de navegação que serão usados para gerar o trajeto desejado, passando pelos objetos que deverão ser visualizados pelo usuário. Na Figura 2.12 o Gerenciador de Trajetos (*VRML Tour Creator*) é inserido no ambiente AWE3D, provendo um recurso adicional ao ambiente adaptativo.

O sistema de ajuda ao usuário, realizado por um agente, é um dos pontos fortes desse trabalho, determinando um roteiro de apresentação em função dos interesses do referido usuário. Este ambiente é adaptável, pois utiliza o perfil do usuário para gerar a trajetória. É estático porque não há modificações do trajeto durante a interação do usuário com o mundo virtual.

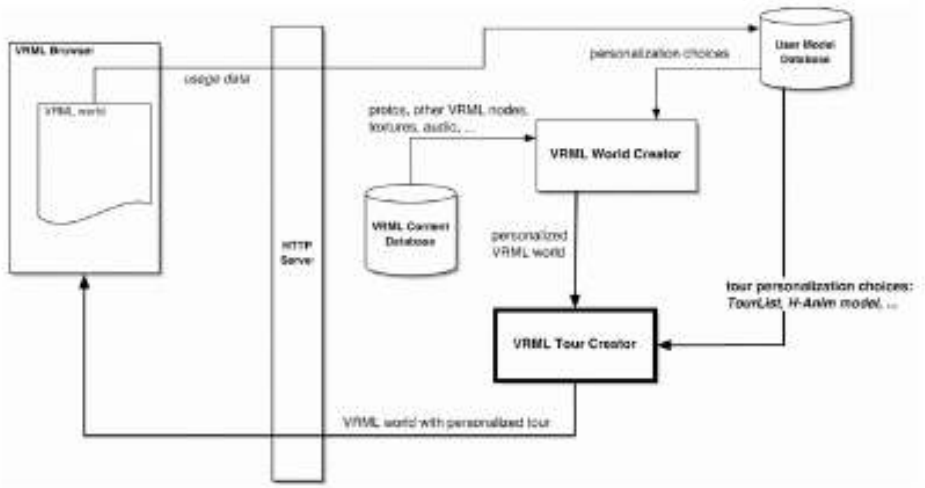


Figura 2.12 Integração do Gerador de Trajetos na arquitetura do AWE3D [Chittaro et al., 2003]

Santos (2004) propõe o ambiente AdapTIVE (*Adaptive Threedimensional Intelligent and Virtual Environment*) que tem a apresentação dos conteúdos adaptados de acordo com os interesses dos usuários e conforme a manipulação (inserção, remoção ou atualização) de conteúdos no ambiente. Um processo de categorização automático é aplicado na criação de modelos de conteúdos, que são usados na organização espacial do ambiente. O processo de adaptação utiliza os modelos de usuário e conteúdo. Além disso, um agente virtual inteligente atua como assistente dos usuários na navegação e na localização de informações relevantes (Figura 2.13).

(a) (b) (c)

Figura 2.13 Atuação de um Agente Virtual Inteligente no AdapTIVE. (a) Solicitação do usuário; (b) localização de uma área; (c) visualização do conteúdo de Inteligência Artificial [Santos e Osório, 2004].

As Figuras 2.13 (a), (b) e (c) ilustram, respectivamente, uma solicitação do usuário para a localização de determinada área e a movimentação do agente; a

localização de uma área pelo agente e o usuário visualizando conteúdos, após clique na descrição do conteúdo correspondente.

A Figura 2.14 apresenta a arquitetura do AdapTIVE. Nela, verifica-se a existência de um agente localizado na máquina do cliente, que acompanha todas as solicitações do usuário no ambiente virtual e o orienta na busca das informações desejadas. O agente no ambiente AdapTIVE possui um papel importante no auxilio à navegação e informa ao sistema, as ações do usuário, tais como, ambientes visitados e conteúdos acessados. Com essas informações, o sistema atualiza o modelo do usuário e o modelo de conteúdos e, em uma sessão posterior (definida pelo sistema), ele reorganiza o ambiente.

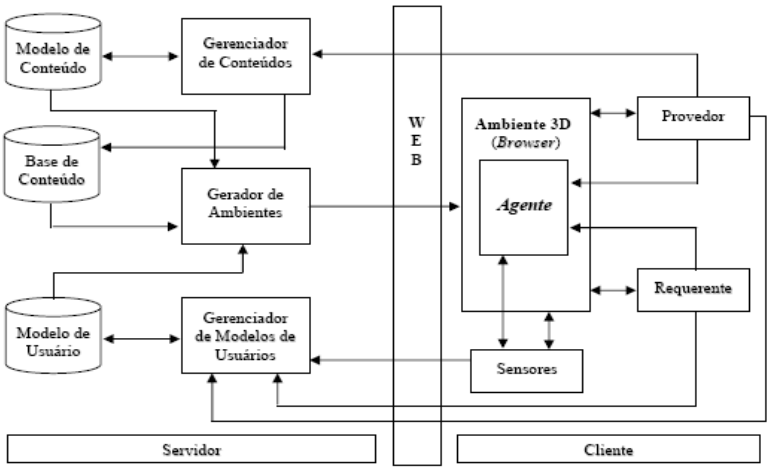


Figura 2.14 Arquitetura do AdapTIVE [Santos, 2004]

O AdapTIVE é um ambiente estático porque não há movimentos de objetos, mas é um ambiente adaptável porque existe a preocupação de adaptação de conteúdo segundo o perfil do usuário.

Estalayo et al. (2004) apresentam uma proposta para realizar a adaptação de conteúdos multimídia em ambientes virtuais, desenvolvida em VRML. Uma aplicação desse trabalho foi desenvolvida com o objetivo de ensinar alunos da marinha na reconstrução de navios. Durante as lições, os estudantes podem percorrer os ambientes selecionados, acessando diferentes tipos de informações multimídias. Estas informações são apresentadas em função de um perfil definido para o usuário. Por exemplo, se a máquina do usuário tem uma placa de vídeo avançada e ele fala espanhol, imagens de

vídeo na língua nativa do usuário são exibidas, auxiliando na complementação do conteúdo em estudo. Caso contrário, apenas imagens textuais são visualizadas.

Na Figura 2.15, é apresentado um exemplo de aplicação do sistema. A vantagem desse trabalho é que não é necessário gerar diferentes descrições para o mesmo mundo virtual, uma para cada perfil. A estratégia é desenvolver um único modelo 3D do ambiente virtual e utilizar uma base de regras capaz de gerenciar a associação e adaptação do processo.

(a) (b) (c) (d)

Figura 2.15 Browser visualizado após a associação da informação de um mesmo modelo 3D. A informação pode ser um vídeo (a), um texto em diferentes línguas (b e d) ou uma página Web (c) [Estalayo et al., 2004]

A arquitetura do sistema é apresentada na Figura 2.16. O *Information Manager* gerencia a inserção dos componentes multimídia nos AV. Ele consulta o perfil do usuário (em *Session Profile Description*) e em seguida acessa o *Information & Interaction Description* que possui um meta-modelo das informações sobre os dados multimídia que estão associados a cada cena. Uma base de dados (*Multimedia Information Database*) é acessada para recuperar esses dados.

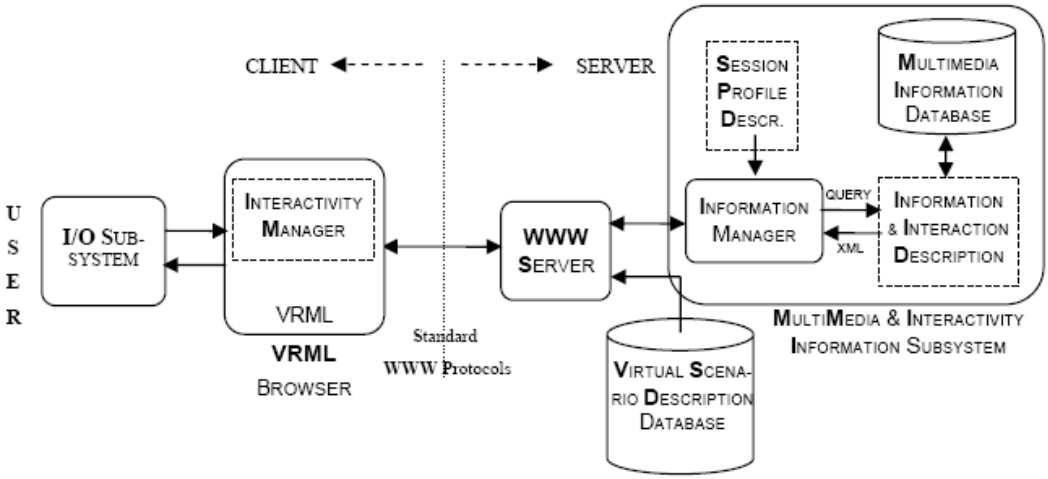


Figura 2.16 Arquitetura para adaptação de conteúdos multimídia em Ambientes Virtuais [Estalayo et al., 2004]

Outra técnica para gerar AV adaptáveis é a apresentação de objetos em níveis progressivos de complexidade. Esta idéia pode ser implementada com aplicações de Realidade Virtual usando LOD (*Levels of Details*).

Kim et al. (2006), por exemplo, usam técnicas LOD para propor níveis de acesso individual para desenvolvedores (projetistas) que se conectam ao ambiente. Essas técnicas atribuem diferentes níveis de prioridade para cada projetista, em função do perfil de cada um. Diferentes níveis de detalhes são apresentados de acordo com o nível de prioridade atribuído. O sistema armazena todos os modelos de visão (e seus respectivos objetos) em um banco de dados e seleciona aqueles que são necessários para construir o ambiente. Na Figura 2.17, o fluxo do processo de gerenciamento dos níveis de autorização é apresentado.

Como trabalhos futuros, os autores propõem a migração do sistema para dar suporte a cenas descritas em X3D e a adaptação automática do perfil em função das ações do usuário durante a interação com o ambiente.

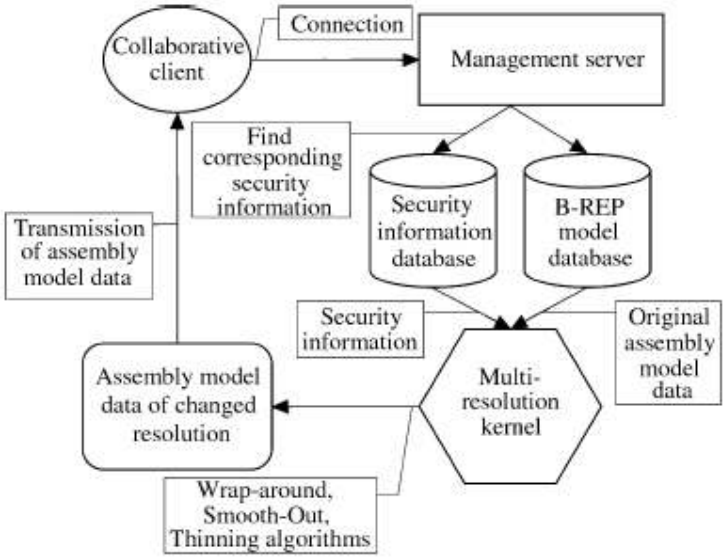


Figura 2.17 Fluxo do processo para um projeto colaborativo com diferentes níveis de prioridade [Kim et al., 2006]

No Quadro 2.2, é apresentada uma comparação entre os Ambientes Virtuais Adaptativos abordados. Verifica-se, neste caso, que os sistemas que possuem o módulo Perfil do Usuário, realizando coleta explícita e/ou implícita, utilizam essas informações para adaptar o ambiente em função das características do usuário.

Quadro 2.2 Comparativo entre os AV Adaptativos

Requisito / Sistema	Tipo de Ambiente	Uso de Agente	Interação Agente/ Usuário	Perfil Usuário	Adaptável (Ambiente)	Tecnologia utilizada
Frery et al. (2002)	Entretenimento, 3D/2D, estático	Repres. gráfica, conhec. do ambiente e do usuário, único	Auxilio a navegação e sugestões	Coleta explícita (formulário), individual, estático	Sim	VRML, Java
Chittaro e Ranon (2002a)	3D, *e-commerce*, dinâmico	Repres. gráfica, conhec. dos objetos, vários	Consulta, auxílio à navegação	Coleta implícita e explícita (formul. e naveg) individual e grupo, dinâmico	Sim	VRML, Java
Chittaro et al. (2003)	3D, entretenimento, estático	Repres. gráfica, conhec. do ambiente, único	Auxílio à navegação	Coleta Explícita (formulário), individual, estático	Sim	VRML, Java
Santos (2004)	3D/2D, disponibilização de conteúdos, estático	Repres. gráfica, conhec. do ambiente, único	Consulta, auxílio à navegação	Coleta explícita e implícita (formul. e naveg) individ., estático	Sim	VRML, Java
Eastalayo et al., 2004	3D/2D, e-learning, estático	Não	Não	Coleta explícita, individual, estático	Sim	VRML, Java
Kim et al., 2006	3D, projetos CAD, estático	Não	Não	Coleta explícita, individual, estático	Sim	VRML, CAD

2.3 Considerações Finais

Com o desenvolvimento das técnicas de RV e o uso de interfaces tridimensionais realistas em ambientes virtuais, o usuário passa a ter maiores possibilidades de interação e navegação em tais ambientes. Verifica-se também, que a RV possibilita maior integração do usuário com a realidade, através do vivenciamento de experiências e de sua participação ativa.

O emprego de Inteligência Artificial em ambientes virtuais (visto na seção 2.1) possibilitou a construção de mundos virtuais mais próximos da realidade do usuário. A aplicação de agentes inteligentes nesses ambientes permite, por exemplo, a presença de assistentes para ajudar na navegação ou servir de interface de comunicação quando o usuário necessita obter alguma informação ou esclarecimento sobre um determinado assunto.

Outra característica importante é que a maioria dos AV estudados utiliza VRML e Java [Java, 2007], demonstrando a importância dessas linguagens para construção de mundos virtuais.

Entretanto, somente com a inclusão do perfil do usuário nos AV, surge a possibilidade de gerar ambientes personalizados, capazes de adaptarem seu conteúdo em função das preferências do usuário, seu estilo de navegação e sua capacidade cognitiva, buscando assim, aumentar a satisfação e produtividade do usuário.

Além disso, um Ambiente Virtual Adaptativo deve ter um agente inteligente responsável pelo monitoramento das ações do usuário e acompanhamento das modificações ocorridas no ambiente, fornecendo informações necessárias ao sistema para atualização do perfil do usuário.

Um dos problemas em um Ambiente Virtual é a evolução desse ambiente em tempo real, permitindo a alteração do mundo à medida que o usuário está interagindo. Os AV Adaptativos estudados realizam suas modificações *off-line*, apresentando-as somente nas próximas interações, não havendo inclusão de novos conteúdos na sessão corrente, mesmo que o usuário tenha aprendido durante a interação. Outro problema identificado nos atuais AV Adaptativos, respondido apenas parcialmente pelo uso de LOD, é a reutilização de mundos, ou parte deles, para compor novos ambientes. O reuso

poderia reaproveitar objetos e conteúdos já criados, proporcionando economia de tempo e de recursos.

No próximo capítulo, é realizado um estudo das técnicas necessárias para a construção do modelo de um Ambiente Virtual Adaptativo, tema desta tese, como parte de sua base conceitual.

3. Infra-estrutura para Representação e Personalização de Ambientes Virtuais

Este capítulo apresenta um estudo das técnicas e tecnologias necessárias para a geração e desenvolvimento de Ambientes Virtuais Adaptativos. Este estudo é necessário para avaliar as melhores formas de representação e armazenamento em AV, além de analisar a infra-estrutura necessária para a integração de tais tecnologias.

Sendo assim, o capítulo 3 foi estruturado em quatro tópicos relacionados à geração e gerenciamento de ambientes adaptativos: Representação de objetos tridimensionais, Gerenciamento de dados XML, Modelagem do Usuário e Agentes inteligentes em AV Adaptativos.

O surgimento da linguagem X3D para representação de AV em substituição à linguagem VRML (ver seção 3.1), abre novas possibilidades de integração de conteúdos 3D com outros conteúdos Web e novas formas de manipulação de objetos em um AV.

Considerando que as novas tecnologias de AV utilizam o padrão XML para representar objetos tridimensionais e que um dos objetivos desta tese é permitir a construção de ambientes virtuais a partir do reuso de objetos armazenados em um banco de dados, a seção 3.2 contém um breve estudo sobre as tecnologias utilizadas para representação, armazenamento e gerenciamento desses objetos em um SGBD.

Na secção 3.2, também é abordada a estrutura de armazenamento de objetos 3D em um SGBD XML nativo, que tem capacidade de representar esquemas XML diretamente no SGBD, e o armazenamento de documentos em SGBD relacionais com suporte a XML, que são capazes de armazenar documentos XML utilizando uma interface para conversão dos dados. Ao final desta seção, é feita uma análise comparativa das vantagens e desvantagens dos SGBD estudados.

Na seção 3.3, é discutida a importância da construção do Modelo do Usuário (MU) para sistemas adaptativos, sendo dada especial atenção ao processo de representação das informações coletadas, definindo as propriedades do modelo e as principais características do usuário. Nela, também é apresentado o uso do MU em

ambientes adaptativos, procurando-se discutir os métodos empregados e as vantagens e desvantagens de cada um. Algumas técnicas aplicadas ao processo de modelagem do usuário também são apresentadas nesta seção com o objetivo de discutir a aquisição de informações e geração do MU.

A seção 3.4 discute o uso de agentes em AV e suas principais vantagens. Esta abordagem é necessária para se poder compreender o comportamento de um agente e determinar a melhor representação para a utilização em um ambiente virtual.

Por fim, na seção 3.5 são apresentadas as conclusões deste capítulo, avaliando-se as vantagens das tecnologias estudas e discutindo-se os gargalos ainda existentes para a construção de AV adaptativos. Tais discussões serviram de subsídios para a construção da arquitetura proposta.

3.1 Representação de Objetos Tridimensionais

Um Ambiente Virtual adaptativo precisa ser construído utilizando objetos que permitam uma interação dinâmica e de fácil atualização das informações do ambiente. A criação de imagens 3D em AV necessita de linguagens de descrição específicas. Tais linguagens devem ser capazes de expressar, de forma precisa, todas as características do objeto que se deseja descrever, tais como forma, textura, cor, escala e posicionamento.

Esta seção tem como objetivo abordar os recursos existentes na linguagem X3D para representação e manipulação de objetos tridimensionais, destacando as vantagens do uso dessa linguagem para a construção de AV.

Objetos tridimensionais possibilitam ao usuário uma noção mais próxima da realidade, pois possuem representação da altura, largura e também da profundidade. Assim, torna-se possível a visualização por diversos ângulos do objeto. Para a representação desses objetos, diversos estímulos visuais podem ser utilizados como, por exemplo, perspectiva (diminuição proporcional dos tamanhos dos objetos à medida que se distanciam do observador); oclusão (objetos mais próximos ocultando os mais distantes, caso os dois estejam na mesma linha de visão do observador); e técnicas de sombreamento (recursos que dão ao observador uma noção da posição do objeto em relação ao solo da cena).

As linguagens de descrição servem para fornecer informações ao computador relativas às imagens que o mesmo deve gerar. Os atributos das imagens estão relacionados à forma, posicionamento, textura dos sólidos, cor, intensidade de luz, transparência, movimento, entre outros.

Diversas linguagens foram desenvolvidas para representar objetos 3D: OpenGL [OpenGL, 2007], OpenSceneGraph [OpenSceneGraph, 2007], Open Inventor [Open Inventor, 2007], DirectX [DirectX, 2007]. Entretanto, algumas delas possuem restrições, seja por terem uma sintaxe complexa, restringindo o seu uso a usuários experientes, ou por ser necessário adquirir licenças de uso.

Muitos avanços foram realizados no desenvolvimento de ambientes virtuais nos últimos anos. O surgimento das linguagens VRML [Ames et al., 1997] e Java3D [Java3D, 2007a] possibilitou a construção de mundos tridimensionais detalhados e interativos na Web.

A linguagem VRML é voltada para construção de mundos virtuais através de objetos armazenáveis em arquivos e requer um suporte completo para poder executar todas as suas funcionalidades. Java3D, por sua vez, por ser direcionada para programas, constrói o mundo a partir de comandos, sendo necessária uma compilação prévia para posterior visualização [Java3D, 2007b]. Além disso, Java3D necessita do ambiente JDK [JDK, 2007] e da linguagem OpenGL [OpenGL, 2007] ou DirectX [DirectX, 2007] para ser executada.

A representação de objetos gráficos tridimensionais utilizando o padrão X3D [X3D, 2007a] aparece como uma solução bastante eficiente para alguns problemas relacionados a AV Adaptativos como, por exemplo, a manipulação de objetos em tempo real durante a interação do usuário ou o reuso de objetos do mundo. X3D é uma evolução de VRML, sendo representada no formato XML. Além de possuir uma padronização mais elaborada, X3D permite uma integração com serviços Web, tornando possível a agregação de novos recursos, o desenvolvimento de ambientes distribuídos, a representação de metadados das informações do grafo de cena e a projeção de novos padrões para aplicações específicas.

O desenvolvimento e a especificação da linguagem X3D estão sob responsabilidade da Web3D Consortium [Web3D, 2007a], tendo sido aprovada pela ISO/IEC em 2006. A especificação da linguagem X3D foi aperfeiçoada em relação a

VRML. Dentre as modificações realizadas, foram introduzidos novos nós e campos de dados para atender os avanços nos dispositivos gráficos comerciais, revisão e unificação do modelo da API (*Application Program Interface*), maior precisão com a iluminação e modelo de eventos e novos formatos de codificação.

X3D emprega uma arquitetura modular para prover maior extensibilidade e flexibilidade. A maioria dos domínios das aplicações não necessita de todos os recursos da linguagem e tão pouco todas as plataformas dão suporte à variedade de funcionalidades definidas na especificação. Sendo assim, dois conceitos foram adicionados: componente (*component*) e perfil (*profile*).

Um componente define uma coleção específica de nós. Tipicamente esta coleção tem em comum um conjunto de funcionalidades (por exemplo, as estruturas NURBS[2] ou as habilidades de texturização). Um perfil é uma coleção de componentes para um nível específico de suporte. Um perfil pode não conter outro perfil, embora ele possa necessitar dos mesmos componentes e níveis do outro perfil, além de vários outros componentes ou níveis. Todos os arquivos X3D requerem a definição do perfil que está em uso. Este perfil pode ser completado com componentes adicionais solicitados pelo usuário [X3D, 2007b].

A Figura 3.1 apresenta a estrutura dos perfis em X3D conforme foi especificada pela Web3D. Foram criados quatro perfis - *Interchange*, *Interactive*, *Immersive* e *Full* - com níveis de funcionalidade crescentes e incrementais. Por exemplo, o perfil *Interchange* é o perfil básico para a comunicação entre aplicações. Este perfil dá suporte aos nós *geometry, texturing, basic lighting* e *animation*. Não existe tempo real para o modelo de renderização, fazendo com que seja muito fácil de usá-lo e integrá-lo em qualquer aplicação. O perfil *Interactive* permite a interação com o ambiente 3D através da adição de vários nós sensores para uso em navegação e interação (*PlaneSensor e TouchSensor, por exemplo),* sincronização e iluminação adicional (*SpotLight, PointLight*). O perfil *Immersive* permite o uso de todos os grafos 3D e interação (Route), incluindo suporte para áudio, colisão, sombreamento de cenas (*fog*) e *scripts*. O perfil *full* inclui todos os nós definidos, inclusive os componentes *NURBS*, *H-Anim* e *GeoSpatial*.

[2] NURBS (*Non-uniform Rational B-Splines*) – componente de X3D utilizado para construir superfícies tridimensionais definidas por funções paramétricas bi-variantes (rosto de uma pessoa, por exemplo).

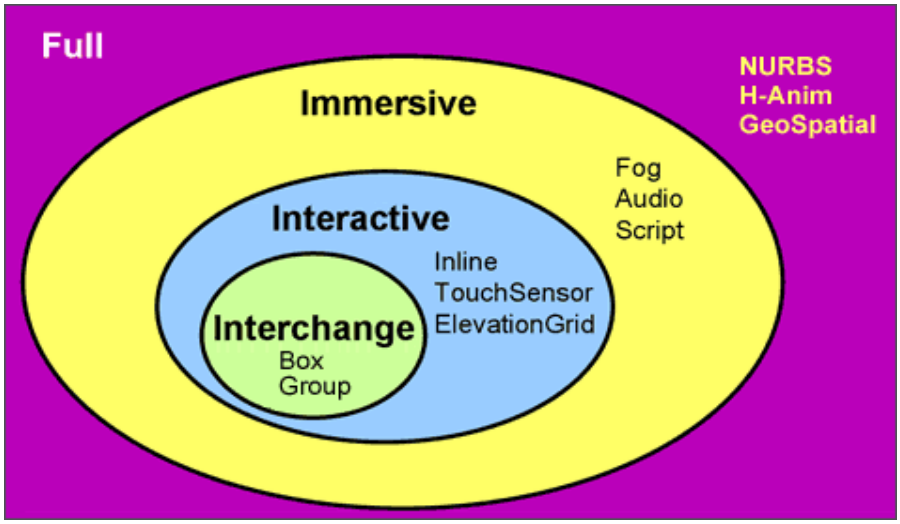


Figura 3.1 Estrutura dos perfis em X3D [X3D, 2007c]

Desta forma, pretende-se reduzir o tamanho dos *plug-in* ao estritamente necessário para executar uma determinada cena, evitando que, por exemplo, arquivos de cenas 3D simples necessitem da instalação de um *plug-in* X3D muito grande.

O mecanismo de componentes de X3D também permite que desenvolvedores implementem suas próprias extensões de acordo com um rigoroso conjunto de regras, evitando a diversidade de extensões que surgiram em VRML97 [VRML, 2007] nos últimos anos, muitas vezes fugindo totalmente das especificações da linguagem.

Portanto, dentre as principais vantagens de X3D, pode-se afirmar que: os aplicativos de edição e de visualização não são obrigados a implementar todas as funções; as comunidades podem implementar suas próprias funções (e.g. geólogos com GeoVRML [GeoVRML, 2007]); funções de visualização evoluídas (NURBS, reflexão, *bumpmapping*) podem ser adicionadas e integradas aos *plug-in* existentes. Além disso, X3D é base para gráficos 3D interativos do MPEG-4 [MPEG, 2007] (televisão digital, aplicações gráficas interativas, multimídia interativa).

3.2 Gerenciamento de Dados XML

Como visto na seção anterior, o uso da linguagem X3D permite construir ambientes tridimensionais utilizando uma estrutura XML. Desta forma, torna-se possível armazenar tais ambientes em um banco de dados capaz de inserir, remover e

alterar dados XML, mantendo os critérios de desempenho e integridade necessários a todo SGBD.

Um estudo das técnicas de armazenamento e gerenciamento de dados XML em SGBD é realizado nesta seção. O objetivo é identificar tecnologias adequadas para a manutenção da consistência dos dados durante o armazenamento e recuperação de objetos 3D, bem como, para permitir a reutilização desses objetos em outros ambientes.

As tecnologias de banco de dados foram adaptadas para garantir o gerenciamento de dados XML de forma eficiente [Mello, 2002]. Sendo assim, surgiram os SGBD XML que possuem um modelo lógico específico para documentos XML.

Um modelo mínimo para dados XML deve conter a definição de elementos, atributos e conteúdo textual (PCDATA), e manter a ordem dos dados no documento [Bourret, 2005]. Um modelo completo deve ainda oferecer suporte à representação de conteúdos mistos de dados, tanto estruturados como semi-estruturados.

Não existe nenhuma restrição sobre o modelo de armazenamento físico a ser adotado em um SGBD XML. Este modelo pode ser tanto um SGBD com formato proprietário de armazenamento XML (SGBD XML nativo) como um SGBD que utiliza um suporte para gerenciar documentos XML (SGBD com suporte a XML). É necessário apenas que exista uma visão lógica de dados no formato XML.

Atualmente, os SGBD disponíveis no mercado não são homogêneos em termos de gerenciamento de dados XML. Entretanto, algumas funcionalidades são consideradas de consenso e encontradas nestes produtos [Bourret, 2005]. Estas funcionalidades são descritas a seguir:

Coleções

Uma coleção é a estrutura a partir da qual os *XML Schemas* [XMLSchema, 2007] relativos a um mesmo domínio de conhecimento devem ser referenciados na base de dados. Desta forma, documentos com estruturas semânticas semelhantes são mantidos em uma mesma coleção. Assim, um SGBD XML pode manipular estes documentos de forma conjunta.

Consultas

Um SGBD XML deve ter pelo menos uma linguagem de consulta. A mais utilizada é *XPath* [Xpath, 2007]. Alguns SGBD dão suporte também a *XQuery*

[XQuery, 2007] e índices definidos sobre dados em coleções, de modo a oferecer um melhor desempenho e expressividade nas consultas. Uma linguagem de consulta para um SGBD XML deve possibilitar desde pesquisas por padrões em textos de documentos até consultas declarativas (típicas de SGBD relacionais) que retornem fragmentos de documentos e possam gerar novas estruturas XML como resposta.

Atualizações

A atualização de dados em documentos XML ainda é restrita aos SGBD XML. Alguns produtos não oferecem este suporte, exigindo que o usuário recupere um documento XML, modifique este documento e o grave novamente no SGBD. Outros oferecem uma interface DOM [DOM. 2007] para acesso e modificação do documento XML, enquanto uma terceira categoria utiliza linguagens proprietárias de atualização de dados, como a *XUpdate* (*XML Update Language*) [XUpdate, 2005]. Esta linguagem permite a inclusão e exclusão de elementos, atributos e texto, assim como a alteração de valores de elementos e atributos.

Gerenciamento de Transações

Um SGBD XML deve dar suporte ao controle de concorrência e *rollback* de transações em caso de falhas. Bloqueios ocorrem geralmente no nível do documento completo, apesar dos modelos lógicos de dados XML permitirem o detalhamento de fragmentos de um documento (elementos e atributos), gerando um baixo nível de concorrência no acesso a dados.

API

SGBD XML oferecem recursos básicos para a comunicação entre aplicações e o banco de dados, baseados geralmente em interfaces ODBC [ODBC, 2007]: métodos para conexão com o banco de dados, execução de consultas e exploração de resultados. Os resultados de consultas são usualmente retornados na forma de *strings* XML ou árvores DOM. Quando múltiplos documentos são retornados, métodos específicos para iteração sobre o conjunto resultante são ativados (semelhante à noção de cursores em SGBD relacionais).

3.2.1 SGBD XML nativo

SGBD XML nativos são feitos especificamente para armazenarem documentos XML. A diferença entre eles e outros SGBD é que seu modelo interno de armazenamento é baseado em XML e não no modelo relacional ou no modelo orientado a objetos. Um SGBD XML nativo possui um modelo lógico para um documento XML e armazena e recupera documentos de acordo com esse modelo.

Como exemplos de SGBD nativos, o Tamino [Tamino, 2007] (proprietário) e o Berkeley DB XML [BerkeleyDB, 2007b] (*open source*) são abordados, sendo destacadas suas características e funcionalidades.

SGBD Tamino

O SGBD Tamino foi o primeiro servidor de banco de dados para dados XML. Ele começou a ser desenvolvido pela empresa Software AG no final da década de 90 e é considerado um dos SGBD XML mais evoluídos existente no mercado, sendo comercializado atualmente na versão 4.4. Tamino é um servidor de XML de alto desempenho para o armazenamento, o gerenciamento, a publicação e o intercâmbio de documentos XML no seu formato nativo, com base em tecnologias de Internet de padrão aberto [Tamino, 2006].

Tamino permite o armazenamento e a manipulação de documentos XML, sendo que cada coleção pode estar associada a um esquema XML para fins de validação de documentos. Os esquemas XML são definidos em um formato proprietário, porém este formato pode ser convertido para *XML Schema Definition* [XSD, 2007]. Elementos ou atributos de um esquema podem ser indexados e existe o suporte a tipos de dados para elementos com conteúdo ou atributos.

Consultas a dados XML podem ser formuladas através de linguagens que são extensões das linguagens *XPath* ou *XQuery*. A linguagem *XQuery* do Tamino possibilita a atualização de dados de documentos.

Diversas funções de um SGBD são encontradas no Tamino: gerenciamento de transações (controle de concorrência e *rollback*), níveis de autorização de acesso, processamento otimizado de consultas e ferramentas para administração de banco de dados. Em termos de acesso, basicamente existem API DOM para Java [JDOM, 2007],

JScript (Microsoft) [JScript, 2007], JavaScript (Netscape) [JavaScript, 2007] e ActiveX [ActiveX, 2007]. Existe também API para acesso SQL através de ODBC e JDBC [JDBC, 2007] ou ainda embutido na linguagem C [Kernighan e Ritchie, 1988]. Tamino pode ser acessado diretamente através de um *browser Web*, via protocolo HTTP.

A arquitetura do Tamino está estruturada basicamente em dois módulos: o *Core Services,* que é o responsável pelo gerenciamento, armazenamento e recuperação de documentos XML, e o *Enabling Services*, que viabiliza a conexão com aplicações e fontes de dados externos (Figura 3.2). Seus principais componentes são:

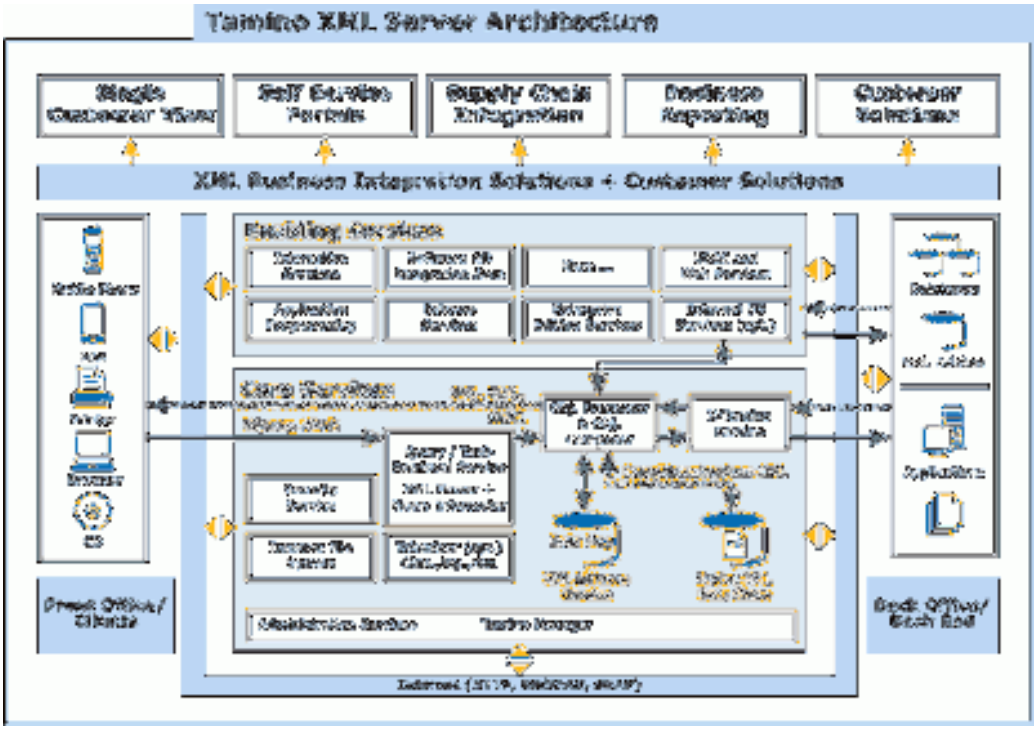


Figura 3.2 Arquitetura do Tamino XML Server [Tamino, 2006]

- *Tamino Manager* é a principal ferramenta de administração do SGBD. Ela é implementada como uma aplicação cliente-servidor, que provê uma interface gráfica para ser executada em ambiente Web;
- *XML Engine* é responsável por armazenar e recuperar objetos XML, baseado nos esquemas definidos no mapa de dados (*Data Map*);
- *Data Map* contém os esquemas que relacionam os objetos XML com o local onde estão armazenados, sendo responsável pela indexação de objetos XML, seja dentro do Tamino seja mapeando os dados de estruturas externas;

- *Query Processor* incorpora expressões *XQuery* capazes de aplicar junções de documentos baseadas em diferentes *doctypes* e realizar consultas e junções em documentos inteiros ou apenas em sub-árvores específicas contidas nesses documentos;
- *Security Manager* pode ser usado para definir ou modificar o privilégio de acesso aos dados armazenados no banco de dados por um usuário ou grupo de usuários;
- *X-Tension* são funções utilizadas para estender as funcionalidades oferecidas pelo Tamino, adicionando uma lógica definida pelo usuário à aplicação. Estas funções tornam possível acessar várias aplicações externas e escrever funcionalidades sob encomenda que permitem ao servidor resolver algumas necessidades específicas da aplicação como, por exemplo, em linguagens de consultas (*Query Functions*), gatilhos (*Trigger Functions*) e funções de mapeamento (*Mapping Functions*);
- *X-Port* é uma interface que conecta o SGBD na Internet via servidor Web sem a necessidade de elaboração de *scripts* ou *servelets*;
- *X-Node* é um componente de integração com outros sistemas, permitindo acessar dados de outros SGBD externos (Oracle [Oracle, 2007a], DB2 [DB2, 2007] e Adabas [Adabas, 2007], por exemplo); e
- *Non-XML Indexer* permite a integração de arquivos não-XML, como por exemplo, documentos do Microsoft Office.

Como linguagem de consulta, o Tamino utiliza um subconjunto de XQL (*XML Query Language*) [XQL, 1999] que usa XML como modelo de dados e é muito similar ao padrão XSL (*Extensible Stylesheet Language*) [XSL, 2007]. No que diz respeito a XSL, que contém informações que permitem converter um documento XML em outro formato, ele processa estilos que estejam sendo usados nos objetos armazenados.

Por ser um SGBD proprietário, existe pouca informação na literatura em relação a uma avaliação do Tamino com outros SGBD (do tipo *benchmark*), como também, a falta de uma análise criteriosa sobre suas vantagens e desvantagens. Contudo, algumas de suas vantagens são:

- Integração e troca de dados – qualquer estrutura de dados interna é representada como XML e, conseqüentemente, é armazenada e tratada como objeto XML.

Mesmo que os dados não tenham sido previamente definidos no Mapa de Dados, o Tamino aceita esses dados, pois supõe que por serem XML, são bem-formados;

- Gerenciamento de dados – capacidade de lidar com grandes volumes de dados e gerenciar processos concorrentes com muita eficiência porque utiliza técnicas como compressão de campos e *cache* de registros;
- Escalabilidade – alto nível de escalabilidade devido ao fato de XML adicionar novos tipos de informação garantindo que os tipos já existentes continuem válidos;
- Segurança – define conceitos de segurança em diferentes níveis, como a camada de aplicação e a camada de transporte, por exemplo. Além disso, dá suporte a muitos dos métodos de criptografia disponíveis no mercado, entre eles *Kerberos* [Kerberos, 2007] e *Secure Sockets Layer* - SSL [SSL, 2007].

Como desvantagem deste SGBD pode-se dizer que: o mecanismo de acesso aos dados é limitado ao uso da API DOM, em conseqüência, há uma demanda maior do uso da memória; a manipulação dos dados deve ser implementada manualmente via programação devido ao fato do acesso ser feito através da API DOM; e a atualização dos documentos só é possível através da substituição de todo o documento.

SGBD Berkeley DB XML

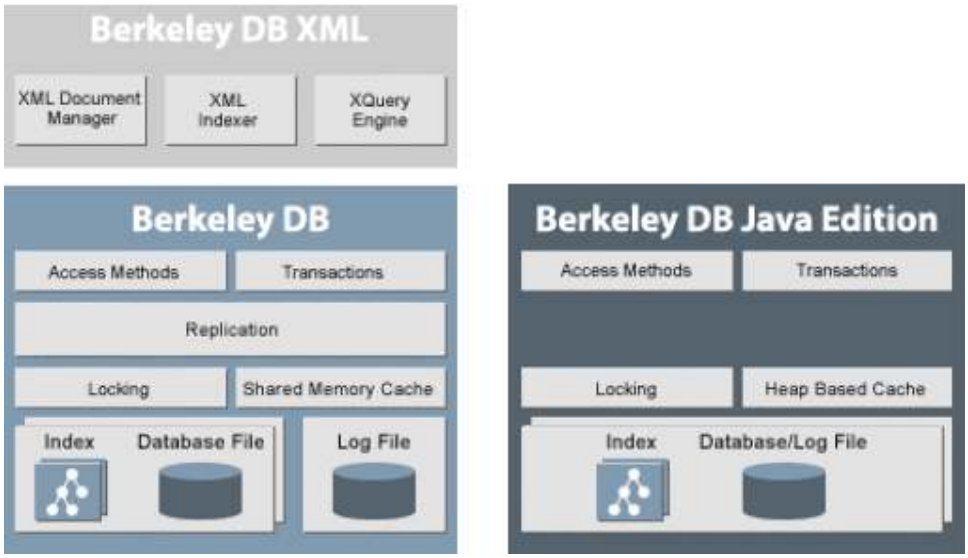
O Berkeley DB XML (BDB XML) é um sistema de banco de dados XML nativo construído sobre uma camada do Berkeley DB [BerkeleyDB, 2007a]. Desta forma, o BDB XML herda a robustez, desempenho e escalabilidade existente no Berkeley DB, bem como dá suporte a transações ACID (atomicidade, consistência, isolamento e durabilidade), Undo e Redo (*write-ahead logging*), utiliza bloqueio para o controle de concorrência e detecta *deadlocks*.

O BDB XML armazena documentos XML em grupos lógicos chamados *containers*, indexados por conteúdo. Estes *containers* representam as mesmas coleções utilizadas em outros bancos de dados XML nativos. Os usuários podem especificar as propriedades por *container*, incluindo a validação de documentos, armazenamento dos documentos inteiros ou como nós individuais, e a seleção dos índices que devem ser criados (elemento, atributo, ou metadado) [BerkeleyDB, 2007b].

O BDB XML tem características robustas que foram testadas em uma grande variedade de aplicações em tempo real. Se uma aplicação usa extensivamente documentos XML, o BDB XML é uma boa solução para a persistência dos dados. Pode ser utilizado como um banco de dados embarcado, conectado diretamente com a aplicação, não sendo necessária a conexão com um servidor e necessitando apenas a manipulação de documentos XML para inserção e recuperação. Este sistema de banco de dados suporta milhares de *threads* simultâneos de processos concorrentes e de controle, manipulando bancos de dados com até 256 terabytes[3].

A arquitetura do BDB XML é apresentada na Figura 3.3. É uma arquitetura simples comparada com outros SGBD (Oracle, p. ex.). Ela possui um gerenciador de documentos XML (*parser* XML), um indexador XML, e a linguagem de consulta XQuery, acoplada ao Berkeley DB.

Além disso, foi gerada uma ferramenta denominada Berkeley DB Java Edition, inteiramente escrita em Java, que dá suporte aos padrões J2EE da Sun Microsystems, com maior flexibilidade e novas otimizações de desempenho, possibilitando que as aplicações se tornem mais rápidas.



(a) (b)

Figura 3.3 Arquitetura do SGBD Berkeley DB XML. (a) Estrutura do suporte a XML (b) Estrutura do banco de dados embarcável para Java [BerkeleyDB, 2006]

[3] 1 terabyte = 1.024 gigabytes

O BDB XML permite o armazenamento de documentos XML, indexação de dados flexível e acesso utilizando XQuery ou XPath, garantindo uma recuperação de dados mais rápida e eficiente. As principais características de armazenamento de documentos são: armazenamento nativo e recuperação de dados XML e não-XML dentro de uma mesma transação; controle de armazenamento flexível por nodos ou no documento inteiro; validação de esquema e suporte a *namespace* de XML.

A linguagem *XQuery* permite acessar documentos XML e expressar relações complexas em consultas que podem ser otimizadas e executadas de forma rápida sob um conjunto grande de dados. As principais características de XQuery são: elaboração de consultas em uma única tabela ou com cruzamento de várias tabelas; possui identificadores permanentes aos documentos para acesso direto; gera otimização da consulta; realiza navegação no conjunto de resultados como no DOM.

O BDB XML é muito útil em aplicações que utilizam XML extensivamente. Além disso, o programa que manipula o banco de dados é livre para decidir como os dados serão armazenados em uma tabela, que pode ter mais de 4 gigabytes de comprimento.

Algumas desvantagens do BDB XML são: não dá suporte a XUpdate e utiliza mecanismos próprios para modificação de dados (falta de um padrão); exige configuração manual para a indexação de nós; não dá suporte a linguagem de consulta SQL, como também não dá suporte ao acesso via rede (tem que usar uma API para isso).

3.2.2 SGBD com suporte a XML

O armazenamento e gerenciamento de documentos XML também têm sido introduzidos em SGBD Relacionais. A forma adotada por muitos fabricantes desses produtos foi a criação de uma interface entre os documentos XML e o armazenamento físico (linhas/colunas) [Siedler, 2004]. Uma das principais vantagens dessa forma de armazenamento XML é permitir o acesso à consagrada e robusta tecnologia de sistema de banco de dados relacionais.

Com a estrutura relacional de armazenamento, mantém-se a possibilidade de realizar consultas complexas através da linguagem SQL, o que garante eficiência na

busca de informações [Browne, 2004]. Nessa forma de armazenamento, as aplicações herdam outras características importantes dos bancos relacionais, como por exemplo, um sofisticado sistema de gerenciamento de transações, bom mecanismo de recuperação e diversas ferramentas que dão suporte a este tipo de tecnologia.

Os SGBD Oracle (*proprietário*) e PostgreSQL (*open source*) foram selecionados como exemplos para serem abordados a seguir, considerando principalmente a robustez desses SGBD e a sua consolidação no mercado.

SGBD Oracle

O SGBD Oracle foi desenvolvido pela Oracle Corporation [Oracle, 2007a] e é um software proprietário, possuindo versões para diversos ambientes operacionais. Sua característica principal é o acesso distribuído, compartilhando os dados entre diversos servidores da rede.

A partir da versão 9i [Oracle9i, 2007], o SGBD Oracle apresenta um novo tipo de objeto, o XMLType, que oferece mecanismos para criar, extrair e indexar dados XML. Desde então, diversas revisões foram realizadas para diminuir a complexidade do gerenciamento de dados XML. Esta evolução gerou o Oracle XML DB, embutido no Oracle Database 10g release 2 [Oracle10g, 2007], que possui alta performance, com tecnologia de armazenamento e recuperação do XML nativo. A base de dados do Oracle utiliza integralmente o modelo dos dados XML da W3C [W3C, 2007] e fornece novos métodos de acesso padrão para navegação e consulta XML [Oracle, 2007b].

Na Figura 3.4, é mostrada a arquitetura do Oracle XML DB com dois componentes principais: o armazenamento das tabelas e visões em XMLType e o repositório Oracle XML DB. Existe ainda um conjunto de serviços associados, protocolos e API para esses dois componentes para dar suporte a numerosos cenários.

O armazenamento XMLType tem um conjunto de componentes essenciais. Quando os esquemas XML são registrados no Oracle XML DB, um conjunto de tabelas default é criado e usado para armazenar instâncias de documentos XML associados com o esquema. Este tipo de dados armazena o documento XML com o auxílio de métodos que permitem criar, extrair e indexar os dados contidos no documento e recuperar *tags* específicas com a utilização de XPath.

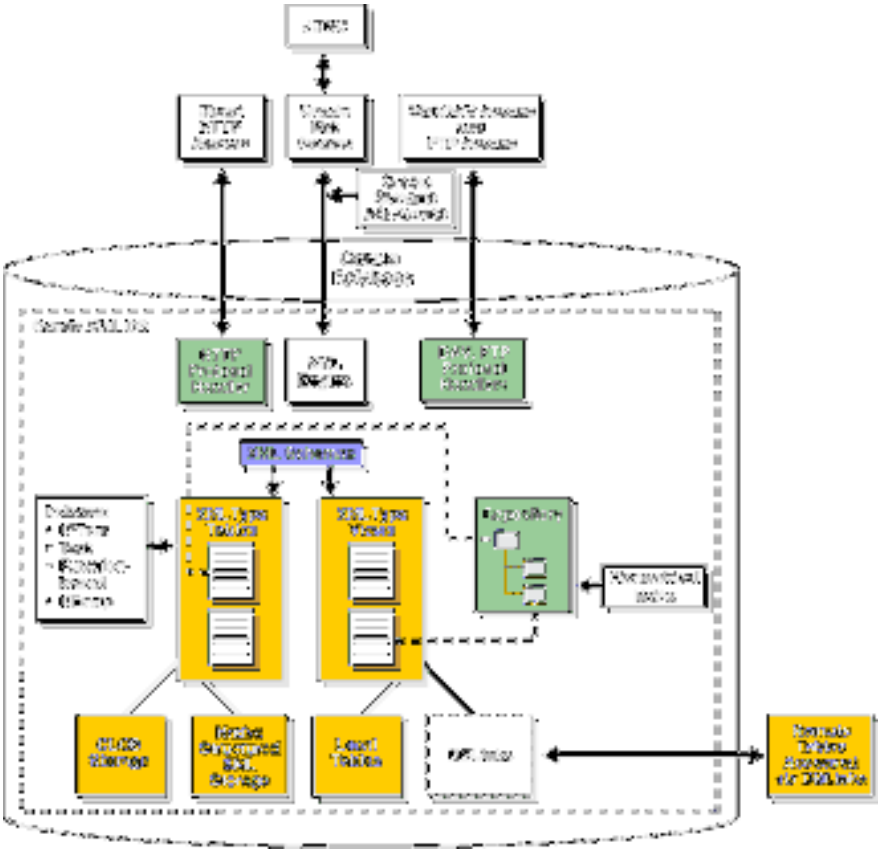


Figura 3.4 Arquitetura do Oracle XML DB [Oracle, 2007c]

Tabelas e colunas XMLType podem ser armazenadas como valores *Character Large Object* (CLOB) ou como um conjunto de objetos. No armazenamento CLOB, o documento XML fica gravado em um arquivo texto na base de dados e há um ponteiro no campo da tabela para esse documento. Uma das maiores dificuldades de se trabalhar com esse tipo de dados é que o documento só pode ser recuperado por completo.

Dentre os benefícios do uso do XMLType, os principais são: permite a integração dos ambientes XML e SQL, tornando possíveis operações SQL em um contexto XML e operações XML em um contexto SQL; possui funções pré-definidas para indexação e navegação; XMLType usa um *parser* e um processador XML embutidos para obter melhor performance e escalabilidade; o XMLType pode ser combinado com outras colunas e tipos de dados nas sentenças SQL. Por exemplo, pode-se fazer uma consulta a colunas XMLType e juntar o resultado com o resultado de uma consulta a uma coluna VARCHAR, sendo que o Oracle pode determinar a melhor maneira de executar essas consultas.

Por outro lado, utilizando o repositório Oracle XML DB, os dados XML podem ser armazenados como um recurso (conjunto de objetos, por exemplo), de forma hierárquica, ao invés de inseri-lo em uma coluna do tipo XMLType. Cada recurso é identificado por um caminho e, além de seu conteúdo, é realizada a persistência de algumas propriedades, como nome, lista de controle de acesso, proprietário e data de criação.

SGBD PostgreSQL

O PostgreSQL [PostgreSQL, 2007a] é um SGBD objeto-relacional com suporte a XML. É versátil, seguro, gratuito e de código aberto. Pode ser encontrado em diversas plataformas, dentre as quais Windows [Windows, 2007], Linux [Linux, 2007] ou qualquer sistema compatível com as especificações POSIX (*Portable Operating System Interface*) [Posix, 2007].

O SGBD PostgreSQL permite a criação de uma base de dados de tamanho ilimitado (a limitação existe apenas em função do tamanho do disco utilizado). Com a versão mais recente (v. 8.2), cada tabela pode ter até 32 terabytes, e cada registro com até 400 gigabytes e cada campo até 1 gigabyte.

Ele tem diversas características presentes em muitos SGBD comerciais como transações, subconsultas, gatilhos, visões, integridade referencial de chave estrangeira e bloqueio (*lock*) sofisticado. Além disso, outras funcionalidades podem ser encontradas como tipos definidos pelo usuário, herança, regras e controle de concorrência de múltiplas versões para reduzir bloqueios (*locks*). Este SGBD também é compatível com uma série de linguagens, tais como PHP, Python, Java, C/C++, Pearl e PL/pgSQL (que é similar a PL/SQL do Oracle) [PostgreSQL, 2006].

Ao contrário de outros SGBD (Oracle, por exemplo), não existe nenhum tipo especial de dados para armazenar documentos XML. Os documentos são armazenados em campos texto, podendo ser acessados diretamente por qualquer comando SQL. As funções de consulta a documentos XML, que são empregadas dentro das cláusulas SELECT, FROM e WHERE, utilizam a linguagem XPath [Sarmento, 2005].

O PostgreSQL contém também uma função para validar um documento XML com uma DTD (*Document Type Definition)* [DTD, 2007]. Este recurso pode ser

utilizado para garantir que todos os documentos armazenados em uma determinada tabela sigam o padrão requerido.

A arquitetura do PostgreSQL pode ser observada na Figura 3.5. Esta arquitetura segue um modelo de comunicação cliente-servidor, em que cada cliente se conecta a um processo servidor. Uma sessão contém diversos processos principais [Khatri et al., 2006]:

- Um processo *postmaster* serve para gerenciar as conexões do usuário. Este processo gerencia também uma área de memória compartilhada para ser utilizada por todos os processos no servidor e por tarefas especiais, tais como, inicializações, paradas e *checkpoints*;
- Um ou mais processos do servidor, denominado *postgres*, são usados pelo *postmaster* para acompanhar as consultas do usuário. Estes processos recebem os comandos enviados pelos clientes e se comunicam uns com os outros utilizando semáforos e memória compartilhada. Assim, conseguem garantir a integridade dos dados, trabalhando simultaneamente em comandos enviados por clientes distintos; e
- Um processo do usuário tal como o *psql* é usado para perguntas interativas do SQL.

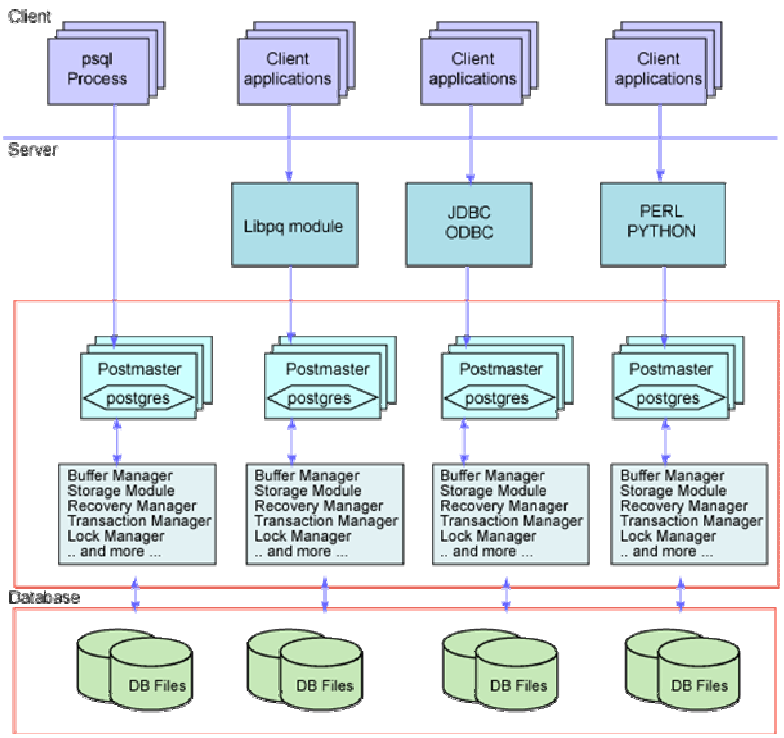


Figura 3.5 Arquitetura do PostgreSQL [Khatri et al., 2006]

Um único processo *postmaster* espera por conexões via TCP/IP (*Transmission Control Protocol / Internet Protocol*) ou *sockets*, e inicia um processo *postgres* para cada conexão. Assim sendo, o PostgreSQL utiliza naturalmente múltiplos processadores, mas para usuários simultâneos e não para acelerar um único comando SQL.

Considerando que o PostgreSQL é um SGBD de código aberto, diversas comunidades em todo o mundo têm desenvolvido aplicações para ele, além de realizarem exaustivos testes durante a sua utilização. Desta forma, ele tem se consolidado no mercado, tornando-se um SGBD estável, robusto, escalável e seguro.

O PostgreSQL possui uma maneira simples de projetar e implementar bancos de dados baseados em servidor. Com uma variedade de métodos, tais como interfaces para Web, acesso a *front-ends* e projeto de interfaces otimizado, este SGBD pode ser usado em um ambiente multi-plataformas [Conrad, 2004].

O SGBD PostgreSQL inclui em sua estrutura muitas funções e sofisticados métodos para manipulação de dados, dando suporte inclusive a sub-consultas SQL. Entretanto, ele não tem implementado a linguagem XQuery, o que reduz as possibilidades de consulta e manipulação de dados em documentos XML.

Por ser uma versão *open source*, o suporte pode ser encontrado em comunidades de desenvolvedores e tutoriais disponibilizados na Web. Contudo, os usuários desse SGBD não têm disponível um sistema de assistência técnica como teriam em uma aplicação comercial.

3.2.3 Avaliação entre SGBD XML nativo e SGBD com suporte XML

Na seção 3.2, foram descritas as funcionalidades para manipulação de documentos XML em SGBD XML nativos e em SGBD com suporte a XML. Particularmente, foram escolhidos sistemas em função da sua capacidade de armazenamento de documentos XML e respectiva consolidação no mercado, sendo considerados tanto sistemas *open source* como comerciais.

No Quadro 3.1, é mostrado um comparativo entre as funcionalidades que os SGBD devem ter para manipular dados XML, de acordo com as características descritas no início da seção 3.2.

Quadro 3.1 Comparativo entre as Funcionalidades de SGBD que manipulam documentos XML

SGBD / Funcionalidade	Tamino	Berkeley DB	Oracle 10g	PostgreSQL 8.0
Coleções	Unidade de armazenemento é o próprio documento, que é agrupado em coleções. Permite o armazenamento de documentos em massa (Data Loader) e a carga através de URL	Documentos podem ser armazenados inteiros ou divididos em nós, permitindo recuperações mais eficientes e alterações parciais de documentos	Armazenamento como uma coluna do tipo XMLType. Pode ser tratado de três formas: CLOB para documentos inteiro; baseado no esquema do documento; e armazenamento híbrido	Armazenamento como uma coluna de uma tabela relacional. Permite a criação de novos tipos, o que pode ser usado para forçar a validação por DTD
Consultas	Extensões de XPath e XQuery. Esta última é utilizada para acesso a múltiplos documentos	XQuery permite expressar relações complexas em consultas com um sofisticado otimizador	Permite o uso de XPath, XQuery e SQL	Permite o uso de XPath e SQL
Atualizações	A linguagem XQuery do Tamino possibilita a atualização de dados de documentos	Permite que sejam feitas atualizações de parte do documento caso ele tenha sido decomposto em pequenas unidades (*sub-trees*)	Permite que sejam feitas atualizações parciais caso o documento esteja relacionado a um XML Schema	Permite atualizar apenas o documento por inteiro
Gerenciamento de Transações	Todos os SGBD realizam controle de concorrência e *rollback* para documentos XML			
API	Interactive Interface, X-Plorer, WebDAV, API DOM para Java, ActiveX, JavaScript	C, C++ e Java, para configurar uma aplicação. Pode ser integrado com o *Apache* tornando-se ideal para sistemas *Web* dinâmicos	HTTP, WebDAV, FTP, PL/SQL e JDBC	JDBC, C^{++}, C#, Perl e Python

Conforme apresentado no referido Quadro, a utilização de um SGBD XML nativo é potencialmente mais promissora no gerenciamento de dados de um ambiente construído em X3D, considerando maior facilidade no armazenamento de estruturas XML. Entretanto, os SGBD relacionais com suporte a XML podem ser uma opção bastante vantajosa, tendo em vista a consagrada robustez e confiabilidade no gerenciamento de dados existente neles.

A escolha do SGBD ideal para dados XML não é evidente. Diversas pesquisas têm sido realizadas no sentido de avaliar as funcionalidades, desempenho e integridade de diversos SGBD com suporte a dados semi-estruturados [Lu et al., 2005; Khatri et al.,

2006; Pires et al., 2006]. Entretanto, cada autor define um SGBD diferente como o melhor ou mais adequado, identificando potencialidades em uns e deficiência em outros. Evidentemente, deve-se ressaltar que a escolha do SGBD está diretamente relacionada com as características e necessidades da aplicação a ser desenvolvida.

No projeto desenvolvido nesta tese (ver capítulo 4), foram utilizados dados semi-estruturados e estruturados. Com o objetivo de avaliar o SGBD mais adequado para gerenciar simultâneamente esses dados, foram realizados testes entre o PostgreSQL e o Berkeley DB. A escolha desses dois SGBD deveu-se principalmente ao fato de serem *open source* e por estarem consolidados no mercado.

Os critérios de avaliação utilizados foram: a forma de tratamento dos dados (estruturados e semi-estruturados); o desempenho na recuperação de dados; e a integridade dos dados. A prioridade nos critérios de escolha é determinada pelo cálculo do risco, tomando por base o impacto e a probabilidade de cada critério ocorrer ($Risco_i = Impacto_i \times Probabilidade_i$).

A ferramenta *JMeter* [JMeter, 2007] foi utilizada neste trabalho para avaliar o desempenho de cada SGBD e as possíveis perdas em inserções de dados. O *JMeter* é uma aplicação criada pelo Apache [Apache.org, 2006], implementada em Java e desenvolvida para realizar testes de carga de aplicações Cliente/Servidor. Com o *JMeter* é possível testar conexões com bancos de dados através do envio de *queries* para um SGBD e avaliar o tempo de execução dessas *queries*.

Esta ferramenta foi escolhida por ser *open source* e simples de simular o uso do servidor de BD, possibilitando antecipar a possível perda de performance e até mesmo a indisponibilidade dos serviços.

O *JMeter* permite gerar vários tipos de relatórios para análise dos testes. Ele dá suporte à representação gráfica das respostas das requisições feitas ao BD e ainda possibilita apresentar dados em formato de planilha, tais como: quantidade de requisições disparadas, tempo de resposta médio, máximo e mínimo das requisições, quantidade de requisições atendidas por segundo, taxa de erro e taxa de transferência de dados em Kb/Seg.

Como resultado dos testes para 100.000 registros, simulando 30 usuários conectados e realizando cada um 20 requisições aos SGBD PostgreSQL e Berkeley DB, obteve-se uma avaliação favorável ao PostgreSQL. Esta avaliação é devida

principalmente a esse SGBD, que tem suporte a XML, ter melhor desempenho no gerenciamento simultâneo de dados estruturados e semi-estruturados. Esta avaliação foi importante também para verificar os critérios de desempenho e de integridade que se demonstraram satisfatórios para ambos os SGBD estudados. Maiores detalhes da metodologia de análise utilizada encontram-se em Albuquerque e Almeida (2006).

3.3 Modelagem do Usuário para Sistemas Adaptativos

O objetivo desta seção é identificar a melhor representação para o Modelo do Usuário (MU) a ser utilizado nesta tese. Para tanto, técnicas de modelagem do usuário e alguns métodos de geração do MU são discutidos nesta seção.

A modelagem do usuário tem tido grande impacto em áreas como filtragem de informação [Herlocker, 2004], sistemas de recomendação de produtos e serviços [Schwab e Kobsa, 2002; Teltzrow e Kobsa, 2004], comércio eletrônico [Alpert et al., 2003], sistemas de auxílio (*help*) [Virvou e Kabassi, 2004], auxílio ao aprendizado *on-line* [Virvou e Chrysafiadi, 2006], auxílio à colaboração entre usuários [Boticario et al., 2005] e agentes de interface [Höppner, 2003].

Um dos principais benefícios da construção de sistemas capazes de modelar o usuário está na possibilidade de adaptar o comportamento do sistema às necessidades particulares de seus usuários [Papatheodorou, 2001].

Apesar das técnicas de modelagem terem sido desenvolvidas para ambientes 2D, a modelagem do usuário também pode ser aplicada em ambientes 3D. Neste caso, algumas características podem ser acrescidas ao modelo, mapeando não somente as ações efetivamente realizadas (seleção de objetos e acesso a conteúdos, por exemplo), mas também a forma como a interação é feita (preferências pela capacidade cognitiva).

Tais informações são armazenadas no Modelo do Usuário e representam o conhecimento do sistema sobre o usuário, sendo utilizado com o propósito de melhorar a interação [Fischer, 2001]. O MU é uma representação explícita de propriedades de um usuário, que permite ao sistema, adaptar diversos aspectos de seu desempenho e de suas funcionalidades às necessidades individuais deste usuário [Rosatelli e Tedesco, 2003].

A seguir, serão discutidas as etapas a serem observadas para construção do modelo do usuário, com o objetivo de identificar as propriedades, características e metodologia necessárias no desenvolvimento do referido modelo.

3.3.1 Processo de Construção do Modelo do Usuário

O processo de modelagem do usuário e a respectiva representação dos dados no modelo devem seguir as seguintes etapas: i) identificar os propósitos de utilização do modelo (pode auxiliar na especificação); ii) definir as propriedades do modelo; iii) definir os dados que deverão ser coletados; iv) determinar o método de coleta dos dados; e v) representar os dados no MU.

Segundo Pohl (1999), um sistema que utiliza o MU como processo de representação do usuário, deve ter como meta os seguintes propósitos: formar suposições sobre os usuários a partir de seu comportamento; representar e armazenar suas suposições; ter habilidade de inferir novas sentenças (ou suposições) a partir das já existentes; ter conhecimento suficiente para lidar com inconsistências entre as suposições; e fornecer as informações constantes do modelo para o sistema.

Uma vez definidas as metas do processo de representação do usuário, faz-se necessário definir as propriedades do Modelo do Usuário. Kass e Finin (1988) descrevem estas propriedades como:

- *Especialização* – o MU pode ser genérico, representando as propriedades e características de um grupo de usuários, ou específico, representando apenas as informações de um individuo;
- *Modificabilidade* – se o MU é modificado durante a interação do usuário, ele é considerado um modelo dinâmico, caso contrário, é denominado estático;
- *Extensão temporal* – um modelo pode abranger os interesses de curto prazo (*short-term*) do usuário, de longo prazo (*long-term*), ou ambos. Os modelos de curto prazo descartam o modelo do usuário assim que a interação termina. Os modelos individuais e os modelos estáticos refletem as propriedades do usuário e, portanto são considerados de longo prazo; e

- *Método de uso* – o MU pode ser descritivo (com dados armazenados em um BD) ou prescritivo (o sistema interpreta os comportamentos do usuário, extraindo informações para compor o modelo).

Outro fator importante para a construção do modelo é definir quais as características do usuário que serão utilizadas, considerando o que será necessário para o processo de adaptação: suas habilidades cognitivas, crenças, preferências e objetivos, por exemplo. Neste sentido, Jameson (1999) propõe os seguintes tipos de dados do usuário que devem ser incluídos no MU:

- Características Pessoais – nome, sexo e idade, por exemplo, são sempre importantes para identificação do usuário. Em ambientes virtuais de ensino, o grau de escolaridade, por exemplo, pode ser importante para a definição do nível de complexidade do conteúdo a ser apresentado;
- Interesses e Preferências – em um sistema de navegação em um AV, conhecer as preferências do usuário sobre apresentação de conteúdos em 3D ou 2D, ou a intensidade de iluminação no ambiente, pode contribuir para uma melhor interação do usuário;
- Habilidades – identificar a capacidade que o usuário tem na utilização de um determinado ambiente ou no conhecimento de um determinado conteúdo, permite ao sistema reduzir algumas etapas na interação com o ambiente ou apresentar informações mais detalhadas sobre o assunto;
- Objetivos – conhecer os objetivos do usuário auxilia o sistema a determinar as inferências necessárias para gerar soluções mais adequadas (por exemplo, se o objetivo é encontrar um trajeto entre os pontos A e B, sem necessariamente ser o mais curto, o sistema poderá propor rotas alternativas para atingir o objetivo do usuário);
- Crenças – o comportamento do sistema adaptativo será influenciado em função do conhecimento que o usuário tem sobre o domínio da aplicação;
- Padrões de Comportamento – identificar o padrão de comportamento do usuário durante a interação é um fator importante, pois ajuda o sistema a definir a sua adaptação, tendo em vista o seu interesse em melhorar a motivação do usuário e tornar a interação mais rica e interessante.

A próxima etapa no processo de construção do MU consiste na escolha do tipo de coleta dos dados. Diversos métodos têm sido apresentados na literatura. Em geral, eles podem ser agrupados em explícitos e implícitos [Papatheodorou, 2001].

A coleta explícita obtém as informações diretamente do usuário, por exemplo, questionando-o sobre os seus interesses, preferências e necessidades. Este tipo de coleta geralmente é feito através do uso de formulários que requisitam informações típicas, tais como sexo e ano de nascimento, e algumas informações específicas, como interesses e preferências (em um sistema de recomendação de produtos, por exemplo, poderão ser solicitadas as categorias de interesse). Em um ambiente virtual de ensino, o nível de conhecimento do usuário sobre o conteúdo, familiaridade com a ferramenta e estilo de aprendizado, são informações que também podem ser obtidas na coleta explícita.

A identificação dos interesses do usuário não deve estar baseada somente na avaliação explícita, mas deve ser adquirida também de forma implícita, através de observações da interação do usuário com o sistema.

A coleta implícita infere informações dos usuários através do monitoramento do seu comportamento durante a interação com o sistema. Em um AV, a coleta dos dados pode considerar para análise a aproximação do usuário a um objeto e a seleção desse objeto, como também os ambientes visitados ou caminhos percorridos. Em ambientes de ensino ou simulação, podem ser incluídas as respostas do usuário ou as decisões tomadas por ele.

Em contraste com os métodos explícitos, os métodos implícitos têm a vantagem de coletar as informações com menor custo, em maior quantidade e sem a necessidade de sobrecarregar o usuário para a obtenção de informações. Entretanto, os métodos implícitos que coletam informações do usuário são mais difíceis de interpretar e são passíveis de ruídos [Joachims et al., 2007].

Em trabalhos tais como o de Chittaro e Ranon (2002a), Santos (2004) e Joachims et al. (2005), os métodos explícitos são utilizados para compor um modelo inicial do usuário e os implícitos aplicados para a atualização do modelo. Desta forma, os métodos implícitos e explícitos são complementares, permitindo obter um conjunto mais completo de informações sobre o usuário.

3.3.2 Modelo do Usuário em Ambientes Adaptativos

Atualmente, os sistemas que interagem com o usuário estão cada vez mais preocupados em adaptar a sua interface e a forma de comunicação de acordo com as preferências e interesses do usuário. Para tanto, tais sistemas utilizam o MU como um mecanismo de auxílio na adaptação do ambiente.

A personalização em aplicações de comércio eletrônico tem se tornado uma importante estratégia de negócio, como por exemplo, serviços de recomendação de produtos direcionados às necessidades do usuário [Distani et al., 2005]. Experiências como a da Amazon [Amazon, 2007] têm permitido que cada consumidor na Internet possua um marketing específico em função de suas preferências.

Em sistemas de recuperação e filtragem de informação, cujo objetivo é selecionar documentos a partir de uma grande quantidade de dados, o MU é utilizado para disponibilizar ao usuário informações importantes ao seu interesse e filtrar as irrelevantes. O MU também é utilizado em softwares educacionais, possibilitando aos sistemas se adaptarem em função das habilidades e conhecimentos do aluno.

Um dos primeiros sistemas a adotar o MU para recomendação de páginas foi LETIZIA [Lieberman, 1995]. Este sistema é um agente que observa o comportamento do usuário durante a navegação pela Web e registra evidências positivas e negativas de interesses. A visita a uma página pode indicar o interesse do usuário pela página. Caso o usuário adicione-a a favoritos, isso implica em forte interesse. A partir destas evidências são feitas análises dos conteúdos das páginas, utilizando o algoritmo TFIDF (*Term Frequency Inverse Document Frequency*) que atribui, para cada termo em um documento, um valor numérico que indica a representatividade do termo para o referido documento. Este valor é computado dividindo a freqüência do termo no documento pela freqüência total do termo em todos os documentos. Uma lista de palavras, com os pesos correspondentes, é gerada como resultado desta análise e armazenada no MU. O modelo é utilizado, então, para a recomendação de *hiperlinks* a partir da página corrente do usuário.

Mesmo sendo uma das primeiras aplicações sobre modelagem do usuário, o sistema LETIZIA utiliza um método eficiente de classificação de conteúdos em função das páginas visitadas pelo usuário. Entretanto, o sistema não se preocupa com a

identificação de outras características, como por exemplo, o conhecimento do usuário sobre o conteúdo visitado e sua experiência anterior.

Castellano et al. (2001) propõem o uso de Redes Neurais para a modelagem do estudante em um sistema hipermídia educativo. As características dos estudantes são organizadas por grupos de interesse (também conhecidos como *cluster* - conjunto de dados com alguma característica ou propriedade comum) utilizando uma rede neural competitiva. As respostas do estudante, fornecidas a partir de um formulário no início de uma sessão, são submetidas à rede neural que irá classificá-lo em função do modelo de grupo de estudantes existente. Esses modelos também foram gerados anteriormente pela rede neural.

A classificação do usuário (incluindo-o em um determinado grupo), proposta por Castellano et al., é realizada apenas com informações explícitas, não considerando o aprendizado do usuário durante a interação com o ambiente.

Abbattista et al. (2002) propõem a extração de modelos de usuários a partir da análise dos conteúdos dos arquivos de *log* de acesso de cada usuário, em um site especializado em livros. A partir desta análise, são extraídas as características usadas como exemplos para um classificador bayesiano que é utilizado para determinar os itens interessantes e não interessantes para o usuário. Este método é útil porque ele gera um modelo probalístico de categorização de livros. Em seguida, realizada a coleta de descrições de livros de diferentes categorias para serem avaliados pelos usuários. Um algoritmo de clusterização é utilizado nesta etapa para inferir padrões de uso de grupos de usuários.

Schwab e Kobsa (2002) propõem uma abordagem para o aprendizado de um modelo de usuário utilizando apenas evidências positivas coletadas implicitamente, em função das observações do comportamento do usuário. Com base nos objetos selecionados pelo usuário (considerados interessantes) são extraídas as características utilizadas para compor o modelo. A recomendação de novos objetos é feita através da abordagem baseada em conteúdo, utilizando o algoritmo *k-nearest neighbor* para caracterizar os objetos de interesse, e da aprendizagem colaborativa, recomendando objetos que usuários com perfis similares selecionaram no passado.

A abordagem de utilizar evidências positivas é interessante, pois, na maioria dos casos, o perfil do usuário reflete o interesse do usuário em objetos ou conteúdos. O

problema é que, muitas vezes, na abordagem implícita, é difícil identificar tais interesses. Em um AV, a abordagem implícita poderá ser utilizada, pois o uso de sensores e o controle da navegação auxiliam na monitoração das ações do usuário. Mesmo assim, uma abordagem explícita também é aconselhada, pois permite criar um MU inicial para o usuário e identificar com mais precisão as evidências positivas durante a abordagem implícita.

Conforme visto nos exemplos anteriormente citados, os esforços na utilização de um modelo de usuário estão concentrados em interfaces 2D. Os trabalhos de Chittaro e Ranon (2000a, 2000b, 2002a, 2002b) propõem a adoção de um modelo de usuário para a personalização de um ambiente 3D. A proposta destes autores é utilizar um modelo de usuário para adaptação da estrutura do layout de uma loja virtual.

Chittaro e Ranon (2000a) desenvolveram a ferramenta ADVIRT (*The Adaptive VR Store*), tendo construído os perfis do cliente utilizando uma combinação das técnicas utilizadas por Ardissono (1999), Joerding (1997) e Joerding (1999). Essa ferramenta atualizava o Modelo do Usuário através dos dados gravados na loja virtual em visitas anteriores realizadas pelo cliente (por exemplo, compras realizadas e número de visitas).

No ambiente ADVIRT, o MU é organizado em três partes:

- *Biosketch* – informações tais como sexo, ano de nascimento, profissão e nível de escolaridade, são obtidas através de um formulário preenchido pelo cliente;
- Aspectos do consumidor – informações relacionadas diretamente com as compras e o comportamento do usuário na loja. Durante cada visita na loja, algumas ações do comprador são monitoradas tais como: produtos observados; produtos selecionados (quando o cliente quer saber mais sobre um produto específico); produtos colocados no carrinho de compras (selecionados para compra posterior); e produtos que foram realmente comprados. Outra informação contida no referido modelo é o *ranking* de produtos interessantes, que lista os produtos de maior relevância para o comprador;
- Aspectos de interação – as informações desta parte do modelo caracterizam o cliente com atributos diretamente relacionados as suas preferências e comportamento no uso da interface do site. Essas informações estão relacionadas à organização da loja (tamanho e estilo da loja, necessidade de assistência), ao número de visitas realizadas e ao gosto por música ambiente (a música a ser tocada pode ser escolhida de acordo com o gênero de música preferido).

O trabalho de Chittaro e Ranon (2000a) é pioneiro na literatura sobre a utilização do MU para adaptação de um ambiente tridimensional. O uso de um formulário e a associação do usuário a um perfil pré-definido contribuem para a personalização do ambiente 3D. As informações do usuário são armazenadas em um objeto, contendo as características pessoais (coleta explícita), perfil de consumo e forma de interação com o ambiente (coleta implícita). Contudo, a atualização do ambiente em função do MU só é realizada após o final de cada sessão. Este sistema não considera as mudanças de interesse do usuário ocorridas durante a sessão que podem influenciar na adaptação do ambiente.

Santos (2004) também utiliza o modelo do usuário para gerar um ambiente virtual adaptativo. A sua proposta é obter informações do perfil do usuário através de um formulário de coleta de dados para composição do modelo inicial. À medida que o usuário interage com o ambiente, as evidências de navegação, solicitação e acesso aos conteúdos, são coletadas e utilizadas no processo de atualização do modelo do usuário. No trabalho proposto por Santos, o MU é atualizado durante a sessão, mas o uso dessas informações para a adaptação do ambiente somente é feito *off-line*.

3.3.3 Técnicas de Modelagem do Usuário

Diversas técnicas de IA podem ser aplicadas no processo de geração do modelo de usuários. Estas técnicas são mecanismos automáticos e inteligentes, responsáveis pela aquisição de dados do usuário e indução de modelos. Dentre elas, pode-se destacar as técnicas de Aprendizagem de Máquina, as técnicas para Reconhecimento de Planos e os Modelos Baseados em Lógica.

Aprendizagem de Máquina

As técnicas de Aprendizagem de Máquina (AM) têm como objetivo auxiliar o sistema a tomar decisões futuras a partir de experiências anteriores. Estas técnicas são utilizadas no processo de modelagem do usuário, principalmente para determinar formas em que os usuários interagem com o sistema. Os dados coletados a partir do comportamento do usuário são utilizados para construir um MU individual [Papatheodorou, 2001; Paliouras et al., 1999] ou para determinar o comportamento de

um usuário em comparação com o comportamento de outros usuários, definidos em um MU geral [Levy e Weld, 2000]. Jameson e Wittig (2001) propõem que seja aplicado um MU geral para cada usuário entrante no sistema. Durante a interação, o modelo iria sendo adaptado ao perfil do usuário, gerando o MU individual.

As técnicas de AM são utilizadas, principalmente, para determinar e induzir padrões comportamentais ou preferências do usuário em relação a um grupo ou comunidade de usuários [Webb et al., 2001]. Essas técnicas são empregadas para a determinação do MU, especialmente em situações em que o usuário realiza uma tarefa que envolve a seleção entre opções pré-definidas, repetidas vezes. Neste caso, as observações do comportamento do usuário gerariam um conjunto de exemplos de treinamento que seria utilizado por um componente de um sistema de AM para prever as ações futuras do usuário.

Schwab e Kobsa (2002) propõem, em um método de AM para aquisição de modelos de usuários, que informações e objetos sejam organizados em classes. As informações obtidas sobre os interesses e desinteresses dos usuários em cada classe são submetidas a um algoritmo de aprendizagem capaz de distinguir objetos ou informações que podem interessar ou não ao usuário. O objetivo da utilização desta técnica é determinar, de forma automática, o que é interessante para o usuário com base em experiências anteriores.

A aplicação de uma técnica de aprendizagem consiste nas etapas de aquisição dos dados do usuário (implícita ou explicitamente) e das fases de treinamento e teste. Os dados coletados durante o treinamento são processados por um algoritmo de aprendizagem, gerando um modelo que representa os dados. A avaliação do desempenho do algoritmo, e conseqüentemente do modelo, é realizada na fase de testes, a partir de um novo conjunto de exemplos. O modelo gerado é usado para representar os interesses dos usuários ou predizer suas ações futuras.

Segundo Zukerman e Albrecht (2001), existem duas abordagens principais adotadas para construir sistemas de recomendação: a baseada em conteúdo e a colaborativa.

A abordagem baseada em conteúdo (ou baseada em características [Koychev e Schwab, 2000]) é utilizada quando o comportamento realizado por um usuário é um indicador, com certa margem de confiabilidade, do seu comportamento futuro. Por

exemplo, em um ambiente de comércio eletrônico, o sistema identifica os produtos que o usuário mais gosta (em função de compras anteriores e informações sobre produtos) e recomenda produtos semelhantes baseados nisso. Os modelos baseados em conteúdo são utilizados nos casos em que o usuário tende a exibir um comportamento próprio e pessoal.

A abordagem colaborativa, também conhecida como Filtragem Colaborativa, é usada quando se pode supor que um usuário tem comportamento semelhante a um grupo de usuários. Os dados de um grupo de usuários são utilizados para fazer previsões sobre um usuário individual. No exemplo do ambiente de comércio eletrônico, a abordagem colaborativa considera que os produtos que um usuário em particular tem preferência são semelhantes aos de um grupo de usuários e irá recomendar outros produtos que esse grupo de usuários gostou. A abordagem colaborativa é útil quando se tenta fazer uma previsão sobre um novo usuário (desde que ele seja enquadrado em um determinado grupo) ou sobre um usuário conhecido em uma nova situação (quando ainda não é possível fazer uma nova previsão em função das informações existentes sobre o usuário).

Reconhecimento de Planos

As técnicas de Reconhecimento de Planos (RP) desempenham um papel fundamental nos modelos de usuário em ambientes com domínios bem definidos. Conhecer os planos e os objetivos do usuário é um aspecto importante para o processo de modelagem do usuário [Carberry, 2001]. Tais técnicas consistem em inferir os objetivos do usuário e os planos utilizados para atingir esses objetivos [Rosatelli e Tedesco, 2003]. A partir das observações do comportamento do usuário, uma seqüência de ações é identificada, os respectivos planos são determinados e as próximas ações são previstas.

Para fazer isso, o sistema possui um conjunto de ações que o usuário pode executar em um determinado domínio e um conjunto de planos que determinam como o usuário executa essas ações. A partir de uma ação do usuário observada, o sistema de inferência de planos constrói uma seqüência de objetivos e de ações que conectam a ação observada a um dos possíveis objetivos do domínio. Isso é obtido através do encadeamento de ações a objetivos atingidos pela ação, desses objetivos a outras ações

para as quais o objetivo é uma pré-condição ou sub-objetivo; dessas ações para os seus respectivos objetivos, e assim por diante.

Um exemplo de sistema que faz o reconhecimento de planos usando Redes Bayesianas são os assistentes do MS Office [MSOffice, 2007], que capturam as relações de incerteza entre os objetivos e preferências do usuário, observações sobre o estado atual da aplicação, representação de seqüências de ações ao longo do tempo e as palavras usadas na consulta do usuário (se esta tiver sido feita). O sistema gera uma distribuição de probabilidades sobre áreas em que o usuário pode precisar de assistência e calcula também a probabilidade de que o usuário deseje que se ofereça alguma assistência. Por exemplo, se um usuário selecionou algumas células no Excel e em seguida pede ajuda ao Assistente, o sistema pode perguntar: "O que você deseja fazer?" e apresentar uma série de ações possíveis ordenadas de acordo com a distribuição de probabilidades para aquele conjunto de ações.

As técnicas de Reconhecimento de Plano são muito importantes para os sistemas adaptativos, principalmente para sistemas de ajuda (*help*) ou que necessitem da compreensão dos planos e objetivos do usuário.

Modelos Baseados em Lógica

Os Modelos Baseados em Lógica (MBL) consideram que o MU corresponde a um conjunto de suas crenças e as representa de maneira diferenciada. Desta forma, inferências sobre o conteúdo dessas crenças podem ser realizadas [Rosatelli e Tedesco, 2003]

Neste caso, os sistemas de modelagem do usuário devem armazenar vários tipos de informações sobre o usuário, como por exemplo, os planos, objetivos, habilidades e preferências do usuário. Por exemplo, se um usuário em um ambiente virtual está tentando mover uma esfera para um compartimento, fechado por uma porta automática, a representação do modelo do usuário *U* é:

```
Objetivo (U, mover_esfera_compartimento)
Sabe (U, mover_objeto)
Sabe (U, abrir_portaAutomática)
Pref (U, ambiente_iluminado)
Habilidade (U, histórico(interação_ambiente))
```

Este tipo de representação permite determinar que o usuário sabe como mover o objeto (clicar o mouse para selecionar o objeto e arrastá-lo, por exemplo), prefere um ambiente iluminado (intensidade da luz pode ser ajustada, se for o caso) e determinar a habilidade que o usuário tem para interagir com o ambiente através da verificação do seu histórico de uso do sistema.

As representações lógicas têm sido bastante utilizadas devido a sua capacidade de precisão e expressividade, além dos mecanismos de inferência estarem bem definidos e serem simples de implementar. Entretanto, essas representações não conseguem lidar com a incerteza. Por exemplo, em um ambiente virtual de ensino, a informação se o aluno sabe ou não um determinado conceito não é suficiente. Ele pode saber com um determinado grau de certeza, o que poderá influir na adaptação do sistema.

Outros formalismos de representação podem ser utilizados com o objetivo de melhor estruturar o conhecimento, como por exemplo, Frames e Redes Semânticas. Nos Modelos Baseados em Lógica, as regras também podem ser associadas a Fatores de Confiança (FC). Tais regras permitem inferir conclusões (hipóteses) com base em antecedentes (evidências). Para cada hipótese, é possível atribuir um FC, o qual representa o grau de crença associado à hipótese [Anjaneyulu, 1997; Nikolopoulos, 1997; Giarratano e Riley, 2004].

3.3.4 Considerações sobre as Técnicas de Modelagem do Usuário

Nesta seção, foram abordadas as técnicas de modelagem do usuário e sua importância em um ambiente adaptativo. No Quadro 3.2, é apresentado um resumo dessas técnicas com suas características, bem como as vantagens e desvantagens de cada uma em relação a sua aplicação em um AV Adaptativo.

As técnicas de Aprendizagem de Máquina permitem coletar dados do comportamento do usuário e construir um MU individual através da identificação de padrões comportamentais ou preferências do usuário. Entretanto, estas técnicas necessitam realizar as fases de treinamento e testes, que irá necessitar de uma grande quantidade de dados iniciais (dependendo da aplicação, isso nem sempre é possível), tornando o processo lento e complexo.

Quadro 3.2 Técnicas de Modelagem do Usuário

Técnica de modelagem	Características	Utilidade em um AV Adaptativo	
		Vantagens	Desvantagens
Aprendizagem de máquina	Utilizada quando usuário realiza mesma tarefa repetidamente.	Pode determinar ações futuras do usuário.	Necessita de treinamento e de uma massa grande inicial de dados; Necessita de dados classificados; Alto custo computacional.
	Informações e objetos são divididos em classes; Utiliza algoritmo de aprendizagem para identificar objetos de interesse; Baseado em experiências anteriores do usuário.	Determina o que é (e o que não é) interessante para o usuário.	
	Conhecimento léxico para auxilio na elaboração do modelo; Uso de ontologias para auxiliar na aquisição das preferências usuário.	Utilizado quando existe pouca informação disponível para determinar as preferências do usuário.	
Reconhecimento de planos	Inferir os objetivos e determinar como a ação do usuário contribui para atingir os objetivos; Conjunto de ações + objetivos implica na geração do plano; Uso de heurística para reduzir número de hipóteses menos importantes.	Reconhecimento dos planos e objetivos do usuário; Pode ser associada com outras abordagens (p. ex., Redes Bayseanas para inferir planos e Aprendizagem de Máquina para construção de biblioteca de planos).	Geração de múltiplas hipóteses, podendo ocorrer uma explosão combinatória; Necessidade de conhecimento sobre o usuário e de informações sobre o domínio.
Modelos baseados em lógica	Crenças e objetivos são guardados em separado e utilizados separadamente para inferências.	Representação lógica tem precisão e expressividade; Tem mecanismos de inferência bem definidos; Simples de implementar; Podem ser adotados graus de certeza em uma crença.	Não lidam com incerteza diretamente.
	Introdução de operadores lógicos para realização de inferências		

As técnicas para Reconhecimento de Planos, que em função do usuário (e seus objetivos), procuram identificar uma seqüência de ações, determinar os planos e prever as próximas ações do usuário, podem não ser adequadas quando forem utilizadas em um AV, tendo em vista o alto grau de liberdade de navegação que o usuário tem nesse ambiente. Para solucionar parte desse problema, Celentano et al. (2004) propõem o reconhecimento de planos em subconjuntos do ambiente.

Os Modelos Baseados em Lógica têm a vantagem de realizarem inferências sobre as crenças que o sistema tem sobre o usuário. Tais modelos podem tornar o sistema mais ágil, aumentando a capacidade de reagir conforme as necessidades e preferências do usuário e otimizando o tempo de resposta em função das suas ações.

Entretanto, estas técnicas não são capazes de representar todo o formalismo necessário, devendo ser associadas a outros mecanismos para aumentar o poder de representatividade do MU, como por exemplo, atribuição de fatores de confiança às regras de inferência.

3.4 Agentes Inteligentes em Ambientes Adaptativos

Ambientes adaptativos, discutidos nesta tese, são ambientes capazes de se adequar às necessidades do usuário ou às suas características. A adaptação implica na modificação dos conteúdos do ambiente em função do perfil do usuário, proporcionando assim, maior interatividade. Esta interatividade deve permitir ao sistema detectar as ações do usuário e modificar instantaneamente o ambiente e as ações sobre ele.

Para o processo de adaptação é necessário um acompanhamento das ações do usuário e uma avaliação na realização de modificações do ambiente. Este processo pode ser realizado por agentes inteligentes capazes de executarem tarefas em função de objetivos a serem atingidos.

A grande vantagem desses agentes é a capacidade de tomar decisões a partir da interação com o ambiente e o usuário (através de sensores) e de análise dos dados coletados, utilizando uma base de conhecimentos sobre o ambiente e os objetivos do usuário. Esta decisão pode ser convertida na execução de uma ação, na identificação de modificações do ambiente ou na realização da comunicação entre o ambiente e o usuário.

Dentre as definições sobre agentes, encontradas na literatura, aquela que retrata com maior precisão a capacidade de um agente em interagir com um ambiente tridimensional para a resolução de um problema ou para auxiliar o usuário é a de Russell e Norvig (2002). Estes autores definem um agente como um sistema capaz de perceber as informações do ambiente onde está inserido, através de sensores, e de reagir através de atuadores.

3.4.1 Propriedades de um Agente

As propriedades de um agente podem ser determinadas em função das ações que ele deve realizar e da sua capacidade de interagir com o ambiente e com outros agentes [Wooldridge e Jennings, 1995]. Desta forma, para se definir tais propriedades, faz-se necessário determinar inicialmente as características do ambiente onde o agente irá atuar.

Nesta tese, o ambiente virtual a ser utilizado possui uma estrutura dinâmica, porém determinística, onde o comportamento das entidades e dos objetos que atuam neste ambiente é conhecido a priori pelo sistema. Sendo assim, as propriedades do ambiente virtual adaptativo, visualizado por um agente, são apresentadas no Quadro 3.3, utilizando a análise feita por Russell e Norvig (2002) sobre os diferentes tipos de ambientes.

Quadro 3.3 Propriedades de um AV Adaptativo visualizado por um agente

PROPRIEDADES	DESCRIÇÃO
Totalmente observável	Sensores do agente conseguem perceber a cada instante todos os aspectos do ambiente relevantes para a escolha da ação.
Estacionário	O ambiente não muda durante o ciclo percepção-raciocínio-ação
Determinista	Pode se prever exatamente o próximo estado do ambiente, apenas a partir da ação executada.
Discreto	Existe um número limitado de percepções e ações distintas no ambiente. Ex.: execução de um experimento (número fixo de possíveis movimentos em cada experimento).
Não-episódico	As ações do episódio atual irão influenciar as ações realizadas nos episódios posteriores. O agente tem que raciocinar para frente.

Uma vez identificado o tipo de ambiente virtual a ser utilizado, as propriedades dos agentes podem ser definidas (Quadro 3.4). Esta caracterização é importante para a concepção do projeto dos agentes que atuam nesta tese, podendo-se determinar o seu comportamento, a forma de comunicação com outros agentes e o nível de complexidade da sua base de conhecimento para o processo de tomada de decisão.

Quadro 3.4 Propriedades dos Agentes para um AV adaptativo

PROPRIEDADES	DESCRIÇÃO
Autonomia	Capacidade de realizar ações com a finalidade de atingir metas e objetivos, sem a interferência do usuário final
Comunicabilidade	Os agentes devem acessar informações sobre o estado atual do ambiente. Isso requer uma habilidade de comunicar-se com os repositórios dessas informações, que podem ser outros agentes
Cooperatividade	Capacidade dos agentes trabalharem juntos para alcançar o objetivo geral através da ajuda mútua
Personalização	Capacidade do agente de aprender com o usuário e adaptar suas ações de acordo com ele
Planejamento	Habilidade de sintetizar e escolher entre diferentes opções de ações desejadas para atingir os objetivos
Pró-atividade ou Orientação ao Objetivo	Não respondem apenas ao ambiente, mas perseguem um objetivo
Reatividade	Respondem de forma oportuna às mudanças no ambiente
Sociabilidade	Interagem com outros agentes (e possivelmente humanos) através de algum tipo de linguagem de comunicação

Além das propriedades dos agentes, Huang et al. (2001) propõe uma classificação de agentes para ambiente Web, agrupando-os em três categorias. Esta classificação também irá auxiliar na especificação do tipo de agente a ser utilizado em um AV:

- 2D ou 3D – os agentes de Web 2D trabalham com os protocolos http, file e ftp. Tais agentes possuem uma interface baseada em texto para seus serviços. PAMELA [Huang et al., 2001] é um agente de Web 2D que ajuda o usuário a organizar o programa de uma conferência. Os agentes de Web 3D são agentes que trabalham com os protocolos para Web 2D e com os de realidade virtual. Agentes de Web 3D são *entidades* que executam funções em ambientes tridimensionais, como por exemplo, guiar um usuário pelo ambiente [Frery et al., 2002], auxiliar um cliente em uma visita a uma loja virtual [Chittaro e Ranon, 2002a] ou simular um jogo de futebol virtual [Huang et al., 2000];
- Cliente ou servidor – uma aplicação de um agente de Web cliente é servir de assistente pessoal para o usuário na utilização do ambiente. Uma aplicação de um agente de Web servidor é servir como um facilitador para os servidores Web apresentarem informações mais inteligentes; e

- Singularidade ou multiplicidade – um agente de Web singular não considera nenhuma interface de comunicação com outros agentes de Web, enquanto um agente de Web múltiplo considera a possibilidade de interação com os outros.

Em Huang et al. (2002), a classificação dos agentes de um ambiente Web pode ser dependente de domínio (agentes *e-commerce*, de recuperação de informação, de busca, assistente de viagens, de comunidade virtual) e dependente de funções (de negociação, de cooperação e de solução de problemas). Esta classificação permite definir algumas características do comportamento do agente, bem como sua funcionalidade e forma de comunicação.

Na próxima seção, é discutida a estrutura de um ambiente multi-agentes, suas propriedades e formas de interação, necessárias para a descrição da cooperação entre agentes.

3.4.2 Sistemas Multi-Agentes

Os Sistemas Multi-agentes (SMA) caracterizam-se pela existência de mais de um agente que interagem entre si de forma autônoma e trabalham em conjunto para executar tarefas, resolver problemas ou satisfazer um determinado objetivo. Esses objetivos podem ser comuns a todos os agentes ou não [Lesser, 1999]. Cada agente pode comunicar ou cooperar com outros agentes quando necessário, com o objetivo de somar os resultados locais para atingir a solução do problema geral [Jennings, 2000].

Para que se consiga tal objetivo, é necessário que se definam bons mecanismos de comunicação (troca de mensagens), cooperação (auxílio) e coordenação (organização) no sistema multi-agentes.

O mecanismo de comunicação é básico para qualquer sistema que precisa interagir. Dois mecanismos bastante utilizados são: a troca direta de mensagens entre agentes e o quadro-negro. No primeiro, os agentes informam suas necessidades a outros agentes via troca direta de mensagens assíncronas, o que pode ser difícil de implementar caso não haja um bom mecanismo de identificação dos agentes no sistema. No segundo, existe um repositório de informações acessível a todos os agentes no qual eles colocam suas necessidades e esperam que algum outro agente do sistema acesse o quadro e

atenda ao pedido. No segundo caso, a existência de muitos agentes no sistema irá aumentar o número de acessos ao quadro-negro, o que poderá inviabilizar a adoção da solução.

O mecanismo de cooperação está relacionado à forma como os agentes expressam suas necessidades para outros agentes a fim de realizar determinada tarefa ou atingir determinado objetivo. Este mecanismo também pode refletir uma estratégia de ação decidida pelo agente, permitindo a realização de negociação.

O mecanismo de coordenação reflete como os agentes estão organizados a fim de realizar um trabalho em conjunto para a execução da tarefa. Dois dos principais mecanismos de coordenação são: o mestre-escravo (*client-server)* e o P2P (*peer-to-peer*) [Bellifemine et al., 2003]. No primeiro, há duas classes de agentes: os mestres ou gerentes e os escravos ou trabalhadores. Os mestres são responsáveis por coordenar os escravos, definindo tarefas e aguardando os resultados. No segundo, todos os agentes do sistema estão sobre um mesmo nível na hierarquia e sabem das tarefas que são responsáveis para desempenhar.

A forma de interação que ocorre entre os agentes é também um fator muito importante na integração destes. As linguagens de comunicação e sua expressividade definem a capacidade de comunicação de cada agente. Ela deve ser universal e partilhada por todos os agentes, ser concisa e ter um número limitado de primitivas de comunicação.

A FIPA (*Foundation for Intelligent Physical Agents*), com o objetivo de padronizar as tecnologias de comunicação entre agentes, definiu um conjunto de performativas (atos comunicativos) para a linguagem de comunicação ACL (*Agent Communication Language*). Sua especificação consiste em um conjunto de mensagens e descrições dos efeitos da mensagem sobre os agentes que a enviam e os que a recebem [FIPA-ACL, 2007]. Os padrões FIPA vêm crescendo em uso e qualidade, sendo utilizado em diversas plataformas de desenvolvimento de agentes compatíveis com eles.

Considerando a necessidade de se construir um Sistema Multi-agentes eficiente para o VEPersonal, objeto desta tese, é abordado na próxima seção o ambiente JADE, descrevendo a sua estrutura e seu funcionamento. Esta ferramenta é bastante utilizada para o desenvolvimento de SMA e servirá de base para a especificação da arquitetura e do comportamento dos agentes do VEPersonal.

3.4.3 JADE – Ambiente de Desenvolvimento de SMA

JADE (*Java Agent DEvelopment framework*) [JADE, 2007] é um ambiente *open source* para desenvolvimento de aplicações baseadas em agentes, conforme as especificações da *Foundation for Intelligent Physical Agents – FIPA* [FIPA, 2007]. JADE simplifica o desenvolvimento de Sistemas Multi-Agentes (SMA) e é implementado na linguagem Java [Java, 2007].

O objetivo de JADE é garantir um padrão de integração de SMA através de um conjunto de agentes e de serviços de sistema, de acordo com as seguintes especificações: serviço de nomes (*naming service*) e de páginas amarelas (*yellow-page service*); transporte de mensagens; serviços de codificação e decodificação de mensagens; e biblioteca de protocolos de interação.

Um agente em JADE é uma classe Java que é executada em uma *thread* específica. Ou seja, cada agente possui sua própria *thread* de controle. Para especificar a funcionalidade de um agente em JADE é necessário definir comportamentos para cada tipo de mensagem recebida.

Neste caso, o agente espera que alguma mensagem chegue a sua pilha de mensagens e, dependendo do conteúdo da mensagem recebida, inicia um comportamento para o tratamento da informação ou requisição contida na mensagem.

JADE oferece facilidades para o desenvolvimento de sistemas multi-agentes que auxiliam o desenvolvedor na implementação de uma plataforma eficiente de agentes. Com esta plataforma, a comunicação, troca de mensagens e outros atributos que um sistema multi-agentes necessita são gerados automaticamente. Além disso, JADE oferece uma variedade de ferramentas de monitoração, gerenciamento e depuração que ajudam tanto no desenvolvimento, quanto na manutenção e suporte a sistemas multi-agentes. Vale ressaltar a grande preocupação que JADE tem em sempre manter os padrões especificados pela FIPA, aumentando o grau de integração e escalabilidade do ambiente em relação a outros SMA.

Na Figura 3.6, é apresentada a arquitetura interna de um agente genérico em JADE. Verifica-se nesta arquitetura que o agente possui vários comportamentos ativos, sendo que cada comportamento (*behavior*) é composto de ações capazes de tratar uma mensagem.

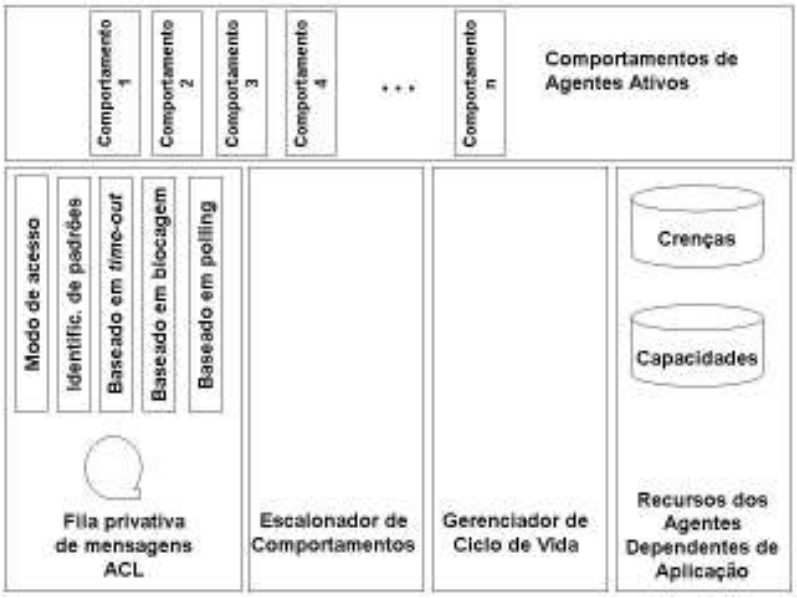


Figura 3.6 Arquitetura de um Agente Genérico em JADE [Pogi, 2007]

As mensagens recebidas são armazenadas em uma pilha, denominada Fila Privativa de Mensagens ACL. O agente é notificado pelo sistema toda vez que uma mensagem é colocada na pilha.

O Escalonador de Comportamentos analisa as mensagens e determina a ordem de execução dos comportamentos necessários para tratar cada tipo de mensagem. O Gerenciador de Ciclo de Vida é utilizado para determinar o estado atual do agente (ativo ou suspenso, por exemplo).

Os Recursos dos Agentes Dependentes de Aplicação armazenam as crenças e capacidades que o agente adquiriu durante a execução de uma ação, podendo ser utilizados na determinação das funcionalidades do agente em função dos serviços solicitados pelos outros agentes.

A comunicação entre agentes é realizada utilizando-se uma mensagem com os seguintes atributos: uma performativa de comunicação que indica o tipo de solicitação requerida (*Request*, *Inform* ou *Query*); o agente remetente; o agente destinatário da mensagem; o protocolo usado na conversação; a referência da ontologia utilizada; o identificador da conversação para controle da comunicação; e o conteúdo da mensagem.

O conteúdo de cada mensagem é regido por uma ontologia, estruturada basicamente em predicados e termos. Na Figura 3.7, é apresentado o modelo de referência de conteúdo de JADE e a seguir, são detalhados alguns de seus elementos.

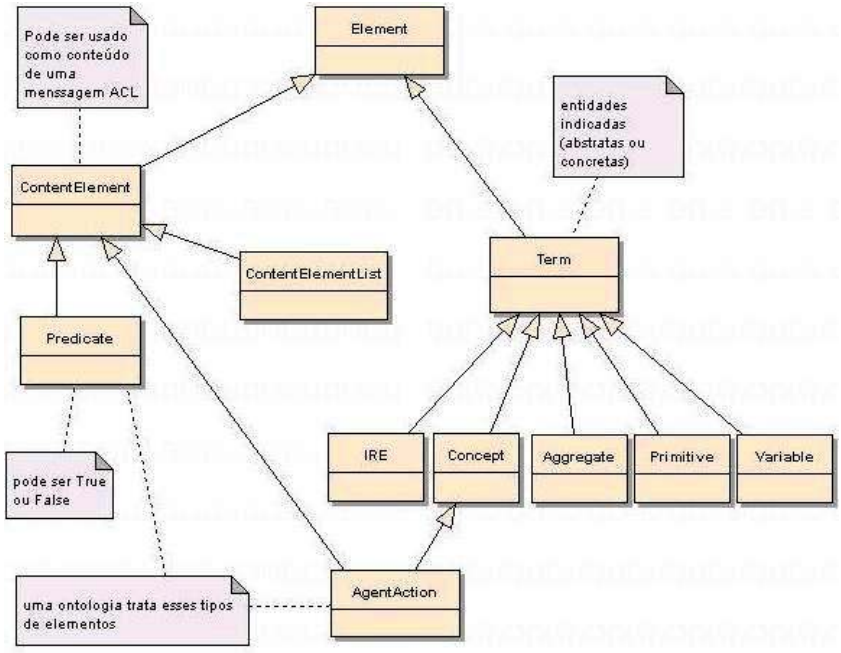


Figura 3.7 Meta-modelo do Conteúdo de JADE [JADE, 2007]

Os Predicados são expressões que têm significado sobre o estado do mundo e pode ser *true* ou *false*. Por exemplo, o sistema para saber se um dado *login* x e uma dada senha y correspondem a algum cadastro de usuário, uma pergunta do tipo "Qual é o *CadastroUsuario* para o qual a senha é *y* e o *login* é *x*?" é construída. Essa relação entre *CadastroUsuario*, *login* e senha é representada pelo predicado *IsLoginCadastrado*. Caso encontre o cadastro solicitado, o agente retorna-o. Caso contrário, ele retorna uma mensagem de falha informando que não foi possível atender à solicitação.

Os termos são expressões que identificam entidades (abstratas ou concretas) que existem no mundo e que os agentes falam e raciocinam sobre. Os termos são estruturados em: *Concepts* – expressões que indicam as entidades; *Actions* – ações que podem ser executadas por um agente; *Primitives* – expressões que indicam uma entidade atômica tal como *strings* e inteiros; *Aggregates* – expressões que indicam entidades que são grupos de outras entidades; *Identifying Referential Expressions* (*IRE*) – expressões que identificam uma entidade para as quais um dado predicado é

verdadeiro; *Variables* – expressões (tipicamente usadas em consultas) que indicam um elemento genérico não conhecido a priori.

3.5 Considerações Finais

Mundos virtuais, construídos a partir de linguagens de representação de objetos 3D, são utilizados para gerar ambientes colaborativos, estimular a capacidade cognitiva e a cooperação entre usuários, mesmo estando remotamente conectados.

Considerando a existência de questões em aberto, principalmente no tocante à utilização de AV dinâmicos e ao acesso via Web, o desenvolvimento de novas tecnologias utilizando o padrão XML aumenta o número de possibilidades de pesquisa.

Além do avanço no desenvolvimento de linguagens de representação, visualização e edição de ambientes 3D, novas tecnologias para SGBD têm sido criadas, permitindo o armazenamento, atualização e recuperação de dados XML com eficiência e precisão significativas.

Com os recursos da linguagem XML, diversas tecnologias para consulta e processamento de objetos 3D contribuem para a construção de mundos virtuais dinâmicos, podendo inclusive recuperar e modificar parte de objetos armazenados em um SGBD, ou mesmo, integrar objetos armazenados em vários SGBD para formar um novo mundo virtual.

Apesar de X3D ainda não ter sido completamente explorada, carecendo de ferramentas de apoio mais interativas para a construção de mundos virtuais, as tecnologias discutidas neste capítulo são ferramentas úteis para o desenvolvimento de ambientes virtuais dinâmicos que estejam remotamente conectados e precisam constantemente modificar o mundo criado, inserindo, alterando e excluindo elementos.

Sendo assim, e conforme discutido neste capítulo, um AV pode ser construído utilizando objetos especificados em X3D e armazenados em um SGBD com suporte a XML. O uso de um SGBD facilita a recuperação de objetos através da elaboração de consultas específicas para geração de um ambiente ou para realização de uma atualização no ambiente existente.

Um grande número de aplicações utiliza o Modelo do Usuário para poder se adequar às necessidades do usuário ou propor mudanças que sejam de seu interesse. Em ambientes 3D, principalmente em AV Adaptativos, o MU torna-se um elemento muito importante, pois ele armazena também o comportamento do usuário, seu estilo de navegação, formas de interação com o ambiente e a evolução do aprendizado do usuário no sistema. Desta forma, o processo de decisão para geração de mundos e eventuais adaptações em função do perfil do usuário, podem ser mais precisas e mais dinâmicas utilizando as informações contidas no MU.

Com a evolução dos ambientes tridimensionais, criaram-se novos desafios para a modelagem do usuário, principalmente na identificação do interesse do usuário em determinados objetos (ou conteúdos) durante a navegação. Chittaro e Ranon (2002a), por exemplo, desenvolveram trabalhos para adaptar um AV utilizando um MU com tais características, o que possibilitou um ganho na representação do modelo e maior personalização do ambiente.

Em um ambiente virtual, a preocupação na representação do MU deve ser pautada na identificação dos objetos que o usuário pode manipular (visualizar, aproximar e selecionar; existindo níveis de percepção diferentes para cada caso), bem como seu interesse por determinados conteúdos, sua capacidade de interação e, principalmente, a sua evolução durante o aprendizado. Para tanto, é necessária a construção de um *framework* que possa armazenar informações específicas sobre o usuário (nível de conhecimento e objetos manipulados, por exemplo) e de um modelo baseado em lógica que interprete as informações necessárias à representação no MU em função dos objetivos e crenças do usuário.

O uso de agentes inteligentes em ambientes virtuais tem o objetivo de aumentar a capacidade de análise do ambiente através da distribuição de tarefas e da incorporação das técnicas de modelagem do usuário na sua base de conhecimentos.

Neste sentido, os agentes inteligentes contribuem para aumentar a capacidade de monitoração das ações do usuário e da identificação do seu perfil. Conseqüentemente, a construção do MU pode ser atualizada com maior acurácia, tendo em vista a capacidade dos agentes em interpretar as "intenções" do usuário.

Entretanto, a adaptação em função do perfil do usuário a partir da geração de conteúdos com níveis de complexidade diferentes ainda é um problema em aberto.

Conforme visto na seção 2.2, a maioria das pesquisas encontradas na literatura se preocupa com a reorganização dos objetos do ambiente ou com a orientação guiada durante a navegação.

Sendo assim, no capítulo 4 será apresentada uma arquitetura para integração das tecnologias descritas neste capítulo como uma solução para a construção de AV Adaptativos. Além disso, uma proposta de representação de níveis de complexidade em objetos tridimensionais também será apresentada como contribuição à adaptatividade de conteúdos para geração de AV.

4. Arquitetura do Sistema VEPersonal para Construção de Ambientes Virtuais Adaptativos

O VEPersonal (*Personalized Virtual Environment*) é um sistema responsável pela criação e manutenção de AV Adaptativos, em que objetos dinâmicos são acessados via Web para navegação e interação. Nesses ambientes, os AV são modificados, em tempo real, de acordo com a evolução cognitiva do usuário [Aquino et al., 2005b].

Os principais beneficiários do VEPersonal são aqueles responsáveis pelo desenvolvimento e construção de AV adaptativos. A infra-estrutura criada pelo VEPersonal permite aos desenvolvedores, gerarem ambientes virtuais compostos por objetos com vários níveis de complexidade. Uma vez definidos os ambientes e as respectivas regras de negócio para funcionamento do ambiente, o VEPersonal é capaz de gerenciar a comunicação com o usuário e a atualização desses ambientes.

Este trabalho é importante para aplicações nas quais diversos usuários, com níveis de conhecimento diferentes, interagem com o ambiente. Neste caso, a modelagem do mundo pode ser a mesma para todos os usuários, bastando incluir níveis de complexidade para os objetos de cada ambiente em função dos perfis dos usuários.

O VEPersonal pode ser utilizado em diversas aplicações tais como ambientes de ensino ou de simulação, em que, à medida que o usuário for realizando tarefas e adquirindo conhecimento sobre o conteúdo, o ambiente poderá evoluir.

Considerando que no VEPersonal os ambientes virtuais devem ser acessados via Web e utilizados por usuários com diferentes *backgrounds* e com capacidades cognitivas distintas, algumas premissas foram elaboradas para a configuração da arquitetura do VEPersonal. São elas: ambiente de baixo custo para ser utilizado via Web; taxa de transferência de dados reduzida entre o VEPersonal e a máquina do usuário; reuso de objetos do AV facilitando a construção de novos mundos virtuais; personalização do ambiente de acordo com o perfil do usuário e da evolução do seu conhecimento durante a navegação; e atualização do AV em tempo real para tornar o ambiente mais amigável e interativo.

Sendo assim, para que os ambientes virtuais sejam executados em plataformas de baixo custo via Web, as aplicações do VEPersonal foram desenvolvidas para computadores *desktop*. O modelo do ambiente virtual utilizado é não-imersivo, não sendo necessário o uso de equipamentos especiais de visualização.

Em relação à taxa de transferência de dados, o sistema VEPersonal foi construído como uma aplicação cliente-servidor. Dada a quantidade de informação que precisa ser manipulada (geração de mundos virtuais, acompanhamento das ações do usuário e modificação de tais mundos), este tipo de aplicação possibilita a redução da carga na máquina do cliente, uma vez que todas as operações mais complexas, que demandam a manipulação de muitos dados ou mais tempo de processamento, são executadas no lado do servidor. O requerimento de largura de banda é também reduzido pelo fato de trafegarem apenas as informações necessárias à aplicação em execução.

A reutilização de objetos para construção de AV foi facilitada com o uso da linguagem X3D (ver seção 3.1) que possibilita o armazenamento de objetos 3D em um SGBD com suporte a XML (seção 3.2) e permite o transporte desses objetos em ambiente Web. Com esta estrutura, é possível construir um *framework* capaz de recuperar apenas os objetos necessários ao AV a ser apresentado ao usuário.

A personalização de mundos virtuais pôde ser realizada a partir da construção do Modelo do Usuário. O emprego de agentes inteligentes permitiu avaliar o conhecimento adquirido pelo usuário durante sua interação com o sistema, baseando-se nas informações armazenadas no Modelo do Usuário, e determinar as alterações necessárias no ambiente.

Finalmente, a atualização do AV em tempo real é realizada por uma interface que permite inserir e remover objetos do mundo virtual sem que seja necessário recarregar todo o ambiente. Assim, o Ambiente Virtual pode ser atualizado durante a interação do usuário, sendo reduzido o volume de dados transferidos pela Web.

Na seção 4.1 é apresentada a arquitetura do VEPersonal, com o objetivo de descrever sua estrutura e funcionalidade, satisfazendo os requisitos descritos acima. Esta arquitetura favorece o acompanhamento da interação e a modificação dos AV conforme as características do usuário e a evolução do seu conhecimento [Aquino et al., 2007].

4.1 Arquitetura do VEPersonal

A arquitetura do VEPersonal é composta por um sistema Multi-agentes e um SGBD, localizados no servidor, e uma interface gráfica (browser 3D) localizada na máquina do cliente. Esta arquitetura, apresentada na Figura 4.1, permite armazenar as características do usuário e seu comportamento no *Modelo do Usuário (MU)*, assim como armazenar o histórico das modificações ocorridas no ambiente no *Modelo do Ambiente (MA)*. Estas informações, obtidas da interface através de sensores que detectam as ações realizadas pelo usuário, são importantes para que o sistema possa acompanhar a sua evolução durante a interação e gerar os objetos do mundo virtual de maneira adequada.

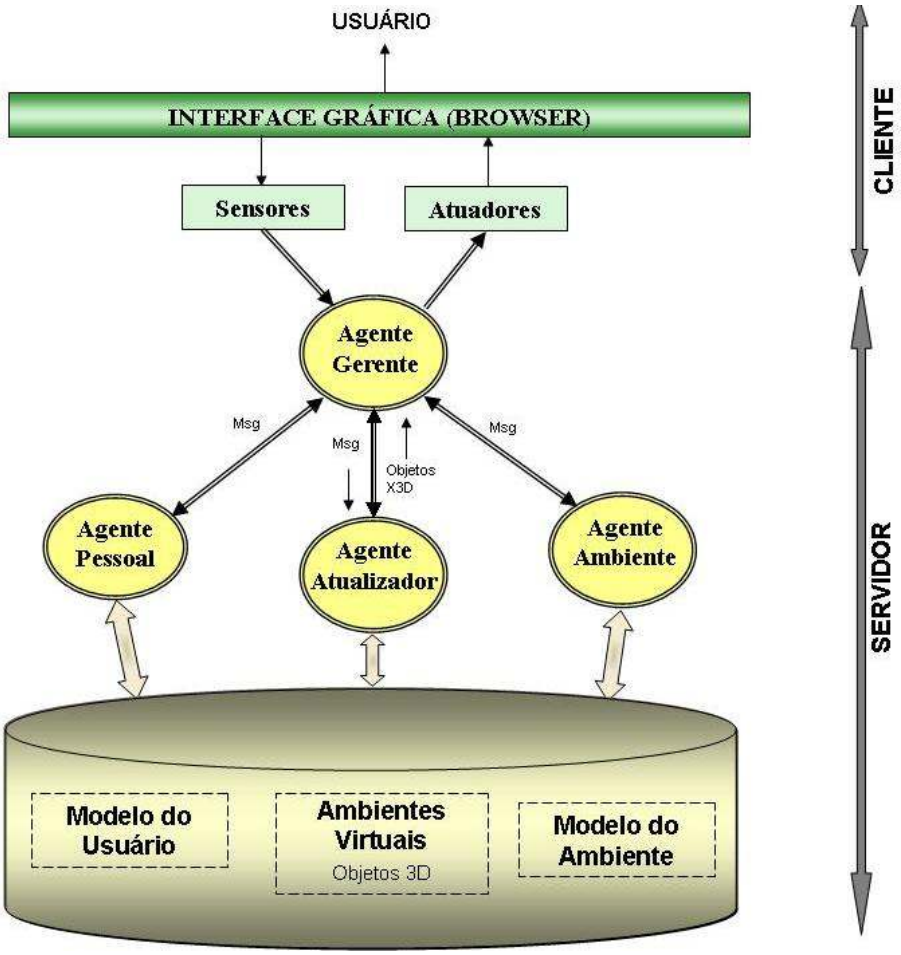


Figura 4.1. Arquitetura do VEPersonal

A análise das informações armazenadas no MU e no MA, e a atualização dos objetos do mundo são realizadas por uma sociedade de agentes. O emprego de agentes

no VEPersonal possibilitou a distribuição das tarefas de acesso ao SGBD e o gerenciamento de vários usuário no sistema. Além dos agentes propostos em Aquino et al. (2005a), um *Agente Gerente* foi introduzido para controlar os outros agentes e facilitar a comunicação com a interface. Assim, quatro agentes são necessários:

- O Agente Gerente responsável pelo acompanhamento das ações do usuário e do estado atual do ambiente virtual. Ele também determina as tarefas para os outros agentes e coordena a comunicação entre eles;
- O Agente Pessoal determina o perfil do usuário e atualiza o MU sempre que alguma alteração do referido perfil for detectada;
- O Agente Ambiente verifica o estado atual do ambiente virtual e armazena as modificações no MA; e
- O Agente Atualizador é responsável pelo processo de atualização do mundo virtual em função das informações fornecidas pelos outros agentes.

Durante o processo de tomada de decisão, o Agente Gerente consulta o Agente Ambiente para obter a estrutura do AV atual (armazenado no MA) e consulta também o Agente Pessoal para identificar o perfil do usuário (armazenado no MU). Estas informações são enviadas ao Agente Atualizador, que gera consultas ao SGBD para recuperar os objetos que deverão ser utilizados na atualização do mundo. De posse desses objetos, o Agente Atualizador os envia para o Agente Gerente que gera a estrutura 3D do ambiente virtual e a remete para a interface de comunicação com o usuário.

Nas subseções seguintes, cada módulo da arquitetura é detalhado, destacando-se sua funcionalidade, importância, tecnologia utilizada e soluções encontradas frente às premissas apontadas no início deste capítulo.

Na seção 4.2, é apresentada a representação dos objetos tridimensionais no VEPersonal, discutindo-se uma proposta para a inclusão de níveis de complexidade em cada objeto. Na seção 4.3, o gerenciamento de dados é discutido a partir do armazenamento de objetos em um SGBD com suporte a XML, considerando a facilidade de recuperação de objetos X3D e do seu reuso.

A seção 4.4 apresenta o Modelo do Usuário utilizado no VEPersonal e o método de modelagem do usuário para identificar o seu perfil. Na seção 4.5, é descrito o Modelo do Ambiente, que foi introduzido na arquitetura para solucionar o controle de atualização dos ambientes virtuais. Então, a seção 4.6 apresenta os agentes a serem utilizados no VEPersonal, bem como o comportamento de cada um e as soluções encontradas para determinar a evolução cognitiva do usuário. Em seguida, a seção 4.7 trata da interface de comunicação cliente/servidor, das tecnologias utilizadas para a comunicação com os agentes via Web e da identificação das ações do usuário.

Finalmente, na seção 4.8, são apresentadas as considerações finais deste capítulo, mostrando-se a importância da arquitetura proposta e justificando as vantagens dessa proposta em função das soluções encontradas.

4.2 Representação de Objetos no VEPersonal

Nesta tese, a proposta de personalização para um Ambiente Virtual, parte do princípio que vários usuários podem visualizar o mesmo mundo virtual, mas com diferentes níveis de detalhamento. Deste modo, um usuário iniciante não é sobrecarregado com informações para as quais não esteja preparado. Já um usuário experiente, que está familiarizado com a aplicação, tem condições de observar com mais atenção os detalhes que antes não poderia perceber.

Durante as pesquisas desenvolvidas para atingir este objetivo, procurou-se determinar um mecanismo capaz de representar um Ambiente Virtual com vários níveis de complexidade. Além disso, era necessário desenvolver um sistema de identificação e recuperação desses níveis para posterior apresentação ao usuário.

Esta proposta não pôde ser aplicada utilizando-se a linguagem VRML, pois ela não permite manipular os objetos de um arquivo (percorrendo os seus nós, por exemplo) para poder recuperar parte das informações contidas nele.

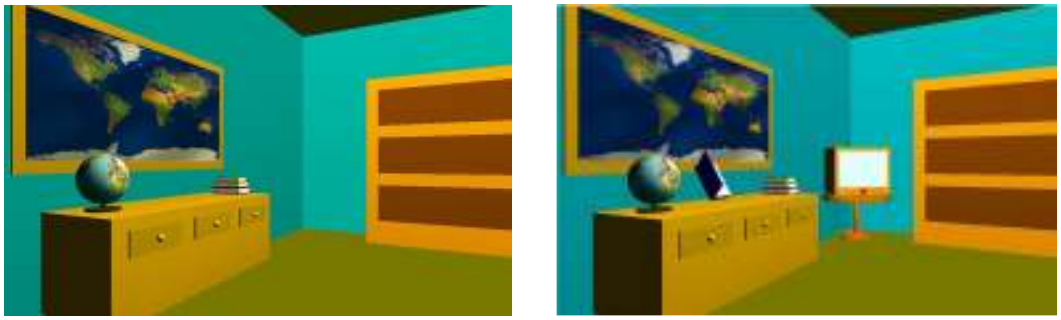
A linguagem X3D surgiu como solução para este problema. X3D é um subconjunto de XML e organiza os objetos do mundo virtual de forma hierárquica, descrevendo cenas tridimensionais como nós contidos dentro de outros nós. (ver seção 3.1). Desta forma, é possível percorrer os objetos de uma cena e realizar, por exemplo, a

inserção ou exclusão de objetos da cena ou a recuperação de um determinado objeto para uso posterior.

A geração de critérios de seleção para visualização do mundo virtual adaptado ao usuário se dá em função das informações contidas no MU (apresentado na seção 3.3) e utiliza técnicas de modelagem do usuário através de um modelo lógico (regras heurísticas) acrescidas de um fator de confiança para determinar o seu perfil (discutido na seção 4.4). De acordo com o perfil, o sistema gera um estereótipo para cada usuário, determinado pelo atributo *userLevel*.

A apresentação de um determinado objeto, com o respectivo nível de complexidade, ocorre somente se este objeto possuir o *userLevel* correspondente ao usuário. Em outras palavras, o *userLevel* identifica quais usuários podem ter acesso a um determinado nó e a seus filhos. Estes atributos são incluídos no código X3D, sem alterar a sintaxe utilizada na representação dos objetos do mundo virtual.

Na Figura 4.2, encontra-se um exemplo de um mundo virtual que apresenta conteúdos diferentes em função do *userLevel*. O ambiente é composto de uma sala constituída de paredes, uma estante e, entre outros objetos, um livro e uma TV. No exemplo, dois usuários, classificados como usuários nível *A* e nível *B*, entram em uma sala. As Figuras 4.2 (a) e 4.2 (b) mostram a visão deles do mesmo ambiente. Verifica-se que na Figura 4.2 (b) aparece um livro e uma televisão na cena, o que não ocorre na Figura 4.2 (a). Neste caso, apenas o usuário nível *B* poderá vê-los, assim como abrir o livro ou ligar a TV, enquanto o usuário nível *A* não toma conhecimento de suas existências.



(a) Sala vista pelo usuário nível A (b) Sala vista pelo usuário nível B

Figura 4.2 Mundo virtual visualizado por diferentes níveis de usuário

A Figura 4.3 apresenta o código representando os objetos do mundo virtual da Figura 4.2. O atributo *<userLevel DEF="B">* determina que os objetos *Livro* e

Televisão (que pertencem a *tag userLevel*) somente serão visualizados pelos usuários que tiverem nível *B*.

```
<Group DEF=”Sala”>
  <Group DEF=”Paredes”.../>
  <Group DEF=”Estante”.../>
  <userLevel Def=”B”>
        <Group DEF=”Livro”>
              <Transform DEF="Capa" ..../>
              <Transform DEF="Contracapa" ..../>
              <Transform DEF="Paginas" ..../>
        </Group>
        <Group DEF=”Televisao”>
              <Transform DEF="MesaTV" ..../>
              <Transform DEF="TV" ..../>
        </Group>
  </userLevel>
</Group>
```

Figura 4.3 Representação dos objetos X3D do mundo virtual da Figura 4.2

Durante o processo de recuperação dos objetos 3D para visualização do ambiente virtual, o *userLevel* do usuário é fornecido. Assim, só são recuperados os objetos que devem ser exibidos ao usuário. Quando é encontrado um objeto que não corresponde ao nível do usuário atual, tal objeto é simplesmente ignorado, como se não fizesse parte da descrição do mundo.

A inserção do atributo *userLevel* em objetos X3D aumenta a possibilidade de representação dos níveis de detalhes dos objetos de um AV. É possível acrescentar o atributo *userLevel* em qualquer parte de um grafo de cena. Isto permite que qualquer nodo de um objeto X3D possa ter mais de uma informação, o que torna possível a inserção de diferentes níveis de detalhes em cada objeto. Por exemplo, em função do perfil do usuário, o quadro na parede da Figura 4.2 (b) pode apresentar um *mapa mundi* político ao invés do mapa topográfico.

A Figura 4.4 apresenta a cena com o mapa político e a Figura 4.5 o código X3D do objeto *Quadro*. A inserção do *userLevel* no nodo *ImageTexture* determina a exibição do mapa político para usuários com nível *“C” (*por exemplo), enquanto que o mapa topográfico é exibido para os que têm nível *“B”*.

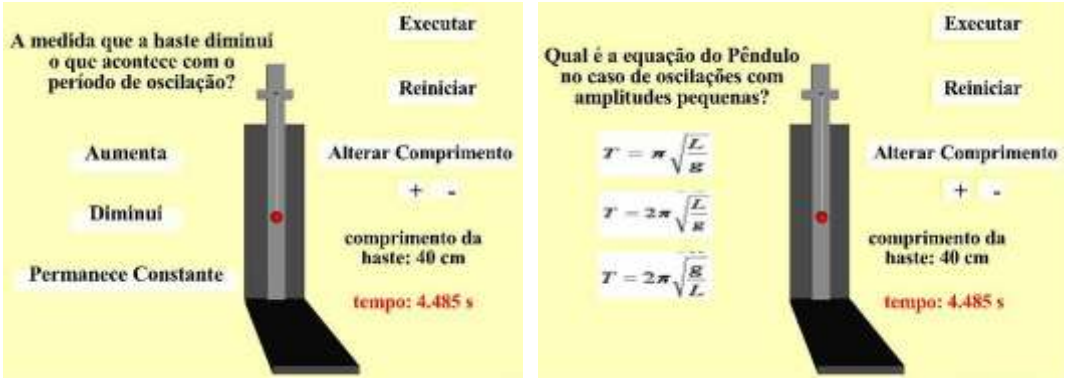
Uma vantagem nesse tipo de representação é que a modelagem do ambiente é a mesma (objetos, posição, movimentos, por exemplo), mas o conteúdo de cada um pode ser modificado em função do nível de conhecimento do usuário.

Figura 4.4 Sala vista pelo usuário nível C

```
<Transform DEF=”Quadro”>
   <Shape DEF=”BordaQuadro” ..../>
   <Shape DEF=”Mapa”>
      <Appearance>
         <Material diffuseColor="0 0 0"/>
         <userLevel DEF="B">
            <ImageTexture url="mapaMundi_topografico.jpg"/>
         </userLevel>
         <userLevel DEF="C">
            <ImageTexture url="mapaMundi_politico.jpg"/>
         </userLevel>
      </Appearance>
   </Shape>
</Transform>
```

Figura 4.5 Representação dos objetos X3D do objeto Quadro

Na Figura 4.6, é apresentado um AV contendo objetos com detalhes diferentes para níveis distintos de usuários. Na Figura 4.6 (a) o usuário tem nível “*Iniciante”* e, portanto, o nível de pergunta do experimento é mais simples que o do usuário *“Avançado”* (Figura 4.6 (b)).



(a) Objetos vistos por usuário nível *“Iniciante”* (b) Objetos vistos por usuário nível *“Avançado”*

Figura 4.6 Visualização do ambiente virtual com níveis de complexidade diferentes

Na Figura 4.7, encontra-se o exemplo do código X3D de um ambiente com as características do AV apresentado na Figura 4.6. Verifica-se que o objeto *Pergunta* é único para ambos os usuários, variando apenas os textos das perguntas e dos botões de respostas (parte do conteúdo do objeto). Isto é possível através da inserção do atributo *userLevel*, podendo ser representado vários níveis de complexidade em um único objeto. O processo de recuperação desses objetos em função do nível do usuário é realizado por agentes e será descrito na seção 4.6.

```
<Transform DEF="Pergunta1Pendulo" translation="-1.3 3 0.5">
   <Shape DEF=”TextoPergunta>
   .......
    <userLevel DEF="A">
       <Text string="A medida que a haste diminui, o que acontece com
          o período de oscilação?"/>
    </userLevel>
    <userLevel DEF="B">
       <Text string="Qual é a equação do Pêndulo no caso de oscilação
          com amplitude pequena?"/>
    </userLevel>
   </Shape>
   <Transform DEF="Resp1P1" translation="-0.2 -1.5 0.1">
    <userLevel DEF="A">
       <Shape>
          <Appearance USE="AparenciaTexto2"/>
          <Text string="Aumenta"/>
       </Shape>
    </userLevel>
    <userLevel DEF="B">
       <Shape>
          <Appearance>
             <ImageTexture url="Equacao1.jpg"/>
          </Appearance>
       </Shape>
    </userLevel>
   </Transform>
   <Transform DEF="Resp2P1" translation="-0.2 -2.3 0.1">
    <userLevel DEF="A">
       <Shape>
          <Text string="Diminui"/>
       </Shape>
    </userLevel>
    <userLevel DEF="B">
       <Shape>
          <Appearance>
             <ImageTexture url="Equacao2.jpg"/>
          </Appearance>
       </Shape>
    </userLevel>
   .....
   </Transform>
</Transform>
```

Figura 4.7 Representação em X3D do objeto *Pergunta* do mundo virtual da Figura 4.6, com diferentes níveis de complexidade

Vale salientar que, para alterar o nível de complexidade do ambiente virtual, não é necessário construir um objeto diferente para cada *userLevel*. Sendo assim, evita-se a duplicação de informação e conseqüente aumento no espaço de armazenamento. Além disso, com o uso do atributo *userLevel* não é necessário um sistema de controle específico para a seleção de objetos.

4.3 Gerenciamento de Dados no VEPersonal

Os ambientes virtuais gerenciados pelo VEPersonal são compostos por objetos 3D que possuem estrutura XML. A construção de tais ambientes é realizada a partir da recuperação dos objetos armazenados em um SGBD. Isto permite a geração de AV adaptativos tendo em vista a possibilidade de recuperar apenas os objetos necessários à montagem de um AV que atenda ao perfil do usuário. Além disso, outras informações são armazenadas no SGBD, tais como o cadastro do usuário e o estado atual do ambiente, que possuem estrutura relacional.

Considerando a necessidade de utilizar um SGBD que gerencie tanto uma estrutura de dados relacional, como uma estrutura de dados XML, o SGBD PostgreSQL (ver seção 3.2.2) foi escolhido porque ele tem suporte a XML; possui uma interface Java para comunicação com aplicações (JDBC), necessária para que os agentes possam interagir com o SGBD; e por ser um sistema *open source*. Além desses requisitos, o PostgreSQL possui robustez e segurança, garantindo a integridade dos dados, e está consolidado no mercado.

No sistema VEPersonal, o modelo do Banco de Dados foi elaborado levando-se em consideração os dados dos usuários, o estado atual do ambiente, os objetos 3D dos ambientes virtuais e os relacionamentos entre esses dados para a construção de um AV adaptado ao usuário. O modelo conceitual, apresentado na Figura 4.8, foi construído utilizando a ferramenta Power Designer [PowerDesigner, 2007]. Este modelo é composto pelas entidades *Ambiente*, *Usuario* e *Objetos*, além dos relacionamentos entre eles.

Devido às constantes adaptações ocorridas na estrutura do banco de dados, durante o desenvolvimento do projeto, optou-se por utilizar o *framework* LiquiBase [LiquiBase, 2007] para realizar o gerenciamento dessas mudanças. Este *framework*

permite a reestruturação (*refactoring*) do banco de dados, bem como a sua atualização, a partir de um arquivo de *log* de mudanças. Este *log* é obtido de um arquivo XML, exportado do Power Designer.

O cadastro do usuário, preenchido na primeira vez que o usuário acessa o sistema, é armazenado na entidade Usuário, juntamente com as informações sobre o perfil do usuário, gerado pelos agentes. Para cada usuário existe um código de identificação (*ID_Usuario*) que é utilizado durante todo o processo de construção e manutenção de um AV (Entidade *Usuario* – Figura 4.8).

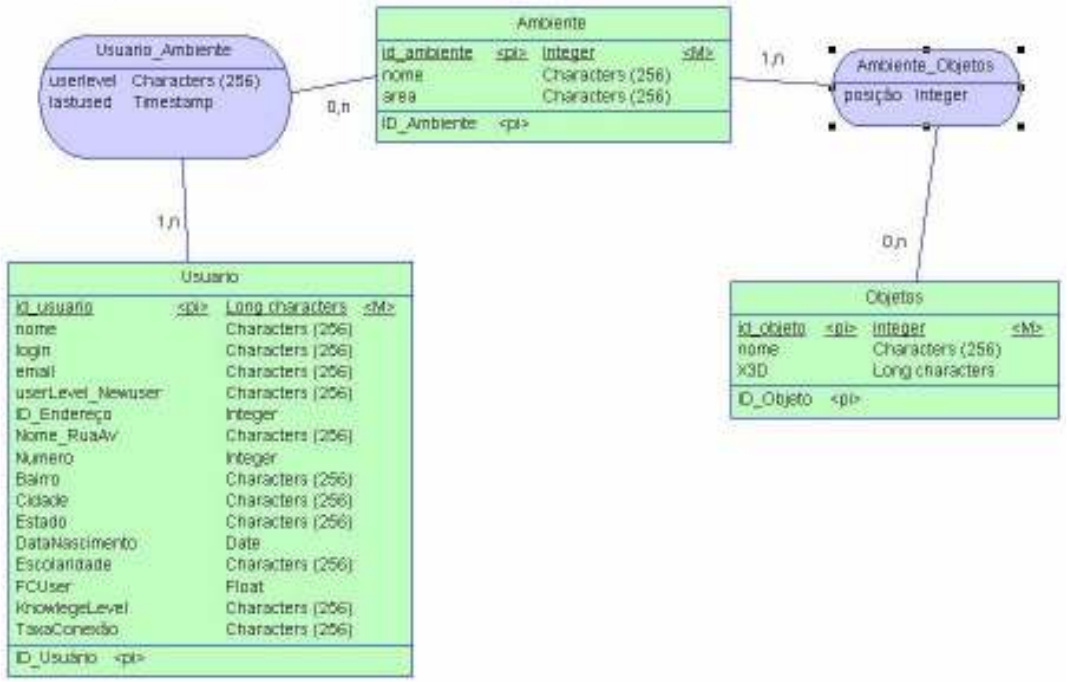


Figura 4.8 Modelo Conceitual do Banco de Dados do VEPersonal

Os objetos 3D são armazenados individualmente no SGBD. Desta forma, é possível a criação de novos objetos e a respectiva inserção no SGBD, bem como a sua reutilização em mais de um ambiente. Cada objeto no SGBD possui um identificador (*ID_Objeto)*, um nome (*Nome_Objeto*) e o respectivo código X3D (Entidade *Objetos* – Figura 4.8).

Um ambiente é composto de um identificador (*ID_Ambiente*), um nome (*Nome_Ambiente*), a área a que pertence (Experimento de Física, por exemplo) e o identificador da área (*ID_Area*) (Entidade *Ambiente* – Figura 4.8).

A relação do ambiente virtual com seus respectivos objetos é gerada pelo sistema a partir da associação do identificador do Ambiente (*ID_Ambiente)* e do nome dos objetos que irão compor o ambiente (*ID_Objeto*).

Na montagem do ambiente, é necessário considerar uma regra em X3D que permite declarar um tipo de um nodo fazendo-se referência a um tipo definido em um outro nodo que já apareceu na declaração do ambiente. Para atender a essa regra, foi necessário criar um atributo para indicar a ordem dos objetos (*posição*) no arquivo X3D. Esse atributo é associado a cada objeto pertencente ao ambiente (ver Relaionamento *Ambiente_Objetos* – Figura 4.8).

Verifica-se, na Figura 4.8, que um ambiente pode ser composto por vários objetos, como também um objeto pode pertencer a vários ambientes. Isto é resultado da propriedade do reuso de objetos em outros ambientes. Sendo assim, os objetos podem ser criados e armazenados independentemente de pertencerem a um Ambiente Virtual.

Toda vez que é necessário gerar um novo Ambiente Virtual para o usuário, o sistema associa um ambiente (*ID_Ambiente*) ao usuário (*ID_Usuario*) com o respectivo *userLevel*, e a data e hora de sua criação (atributo *lastused*) Desta forma, é possível identificar o último ambiente gerado para cada usuário, bem como, o histórico da geração desses ambientes. (ver Relacionamento *Usuario_Ambiente* – Figura 4.8).

Caso seja uma atualização do AV (devido à mudança do perfil do usuário ou acesso a um outro mundo virtual, por exemplo), um novo ambiente (*ID_Ambiente*) é instanciado com os nomes dos objetos atualizados. O *ID_Ambiente* gerado é associado ao usuário. Neste momento, é possível inserir novos objetos nesse ambiente. A decisão de inserir objetos no ambiente é de responsabilidade dos agentes e está descrita na sua base de conhecimentos (ver seção 4.5).

4.4 Modelagem do Usuário

No processo de modelagem do usuário, a coleta de dados é tanto explícita como implícita, conforme comentado no capítulo 3 (seção 3.3). No método explícito, um formulário inicial (Figura 4.9) é apresentado ao usuário na sua primeira visita ao VEPersonal.

Figura 4.9 Estrutura do Cadastro do Usuário

Neste formulário, o usuário fornece informações sobre seus interesses, nível de conhecimento e necessidades, que serão utilizadas para a construção do perfil do usuário. As informações solicitadas no formulário são: características pessoais (nome, sexo, endereço, data de nascimento) e aspectos de preferência de interação (necessidade de assistência na navegação ou na execução da tarefa, taxas de conexão e capacidade de

processamento, por exemplo). Também existem algumas informações desenvolvidas para uma aplicação específica (Experimento de Física) tais como: nível de escolaridade e conhecimento do conteúdo a ser estudado.

O método implícito tem por objetivo obter informações não fornecidas no formulário ou que só podem ser obtidas com a navegação do usuário no ambiente, como por exemplo, objeto visitado ou selecionado, interesse em determinados ambientes e conhecimento adquirido na interação. Este conhecimento pode ser identificado, por exemplo, através da aplicação de testes de avaliação ou verificando se o usuário conseguiu executar uma determinada tarefa com sucesso após tentativas anteriores.

Como visto na seção 3.3, as informações sobre o usuário, relevantes à identificação do seu perfil, são armazenadas no MU. Através dessas informações, o sistema pode acompanhar a evolução do usuário e gerar inferências para a adaptação do ambiente virtual. O MU tem as seguintes propriedades:

- A especialização é individual, definida para cada usuário;
- O modelo é dinâmico, podendo ser alterado durante a interação;
- A atualização do MU é realizada utilizando os modelos baseados em lógica, que consideram o conjunto das crenças, preferências, interesses e desejos do usuário. Um conjunto de regras associadas a Fatores de Confiança [Giarratano e Riley, 2004] é utilizado para determinar o nível de conhecimento do usuário e a atualização do MU.

O perfil do usuário é caracterizado por um atributo denominado *userLevel*. Este perfil identifica o nível de conhecimento do usuário sobre o mundo e é determinado a priori pela coleta explícita realizada na primeira interação do usuário com o ambiente.

À medida que o usuário interage com o ambiente, realizando atividades (respondendo a testes em um ambiente de ensino, por exemplo), o *userLevel* pode ser modificado caso seja identificada uma evolução do seu nível de conhecimento. De acordo com a alteração do *userLevel*, o sistema pode determinar se devem ser inseridos novos objetos no ambiente ou atualizados os já existentes [Aquino et al., 2006].

A seguir, são apresentados os métodos para definição do perfil do usuário, bem como as regras para a sua atualização.

Métodos de Definição de Perfil

O modelo utilizado para determinar o perfil do usuário está baseado em um conjunto de regras do tipo "SE **E**vidências ENTÃO **H**ipóteses" associadas a Fatores de Confiança (FC). Para cada hipótese, é possível atribuir um FC, o qual representa o grau de crença associado à hipótese. Os valores do FC variam entre -1 (descrença total em uma hipótese) e +1 (crença total em uma hipótese) [Anjaneyulu, 1997; Nikolopoulos, 1997].

No VEPersonal, a atualização do perfil é controlada em função da variação do FC do usuário (FC_{user}), calculado inicialmente na coleta explícita e atualizado durante a interação do usuário com o sistema. O FC_{user} tem valor inicial de 0.0 (atribuído antes do usuário responder ao formulário).

Uma aplicação para ensinar Física a alunos do 2º grau será utilizada para ilustrar o uso do FC no cálculo do perfil. Neste caso, as evidências podem ser relacionadas ao nível de escolaridade do usuário (regra R1), sua faixa etária (regra R2) e o seu nível de conhecimento em um conteúdo específico de Física (regra R3). Estas regras são utilizadas para avaliar as informações da coleta explícita. Em função da resposta do usuário, um FC é associado à hipótese de cada regra[4].

Na regra R1, o nível de escolaridade mais alto supõe que o aluno já tenha aprendido determinados conteúdos e realizado tarefas e exercícios durante o ano escolar. Isto é importante para a aprendizagem do conteúdo da aplicação.

R1: Definição do nível de escolaridade do usuário

```
Se aluno está no 1° ano do ensino médio então
  Tem nível de escolaridade para executar o experimento com FC = x
Se aluno está no 2° ano do ensino médio então
  Tem nível de escolaridade para executar o experimento com FC = x
Se aluno está no 3° ano do ensino médio então
  Tem nível de escolaridade para executar o experimento com FC = x
```

Na regra R2, a idade influencia na capacidade de raciocínio, maturidade intelectual e percepção do mundo. Sendo assim, o FC pode aumentar em função da idade.

[4] O valor do FC para cada regra desta seção está representado por "x" e varia entre -1.0 e +1.0.

R2: Definição da faixa etária do usuário

```
Se faixa etária < 15 anos
      Tem idade adequada para realizar aplicação FC = x
Se 15 ≤ faixa etária < 17 anos
      Tem idade adequada para realizar aplicação FC = x
Se faixa etária ≥ 17 anos
      Tem idade adequada para realizar aplicação FC = x
```

Na regra R3, o usuário deve informar no formulário, durante a coleta explícita, qual o nível de conhecimento que ele tem sobre o conteúdo a ser estudado.

R3: Definição do nível de conhecimento do usuário sobre o conteúdo

```
Se o conhecimento sobre o conteúdo = "pouco" então
      Tem conhecimento adequado para realizar a aplicação FC = x
Se o conhecimento sobre o conteúdo = "razoável" então
      Tem conhecimento adequado para realizar a aplicação FC = x
Se o conhecimento sobre o conteúdo = "muito" então
      Tem conhecimento adequado para realizar a aplicação FC = x
```

Para cada FC gerado pelas regras da coleta explícita, o FC do usuário deve ser atualizado. A regra R4 é utilizada para calcular o novo valor do FC_{user}. Na equação dessa regra o novo valor é indicado por FC_{new}, que é baseado no seu valor antigo (FC_{old}) e no FC da regra que está sendo executada (FC).

R4 – Determinação do novo Fator de Confiança

$FC_{new} = FC_{old} + FC\ (1 - FC_{old})$, para ($FC_{old}$ e $FC > 0$) (1)

$FC_{new} = FC_{old} + FC\ (1 + FC_{old})$, para ($FC_{old}$ e $FC < 0$) (2)

$FC_{new} = (FC_{old} + FC) / (1 - \min(|FC_{old}|, |FC|))$, caso contrário (3)

Terminada a coleta explícita, o FC_{user}, que será utilizado para determinar o perfil do usuário, recebe o valor de FC_{new}. A regra R5 classifica o usuário em três níveis: "iniciante", "intermediário" e "avançado". Essa classificação foi utilizada considerando que um usuário que tem pouco conhecimento ou pouca experiência sobre o assunto é considerado "iniciante". Os outros dois níveis, "intermediário" e "avançado", são

atribuídos para os usuários que têm um pouco mais de conhecimento sobre o assunto ("intermediário") e para àqueles que têm muito conhecimento ("avançado"), respectivamente. Em função do nível de conhecimento do usuário é que será definido o grau de dificuldade dos conteúdos a serem abordados no ensino de Física, objeto do estudo de caso desta tese.

Outras aplicações poderiam se basear em uma classificação diferente como, por exemplo, taxa de transmissão da conexão via Web ou permissão de acesso em função de uma determinada hierarquia.

R5: Definição do *userLevel* em função do nível de escolaridade, faixa etária e nível de conhecimento.

/* Os valores para o enquadramento do usuário em um determinado nível (representado pelo *userLevel)* variam entre -1.0 e 1.0 ($-1.0 \leq w, u \leq 1.0$), e podem ser ajustados em função da aplicação e dos atributos a serem avaliados. */

Se $FC_{user} \leq w$ então
    userLevel = "iniciante"
Se $w < FC_{user} \leq u$ então
    userLevel = "intermediario"
Se $FC_{user} > u$ então
    userLevel = "avançado"

As regras de atualização do modelo, durante a coleta implícita, utilizam o FC para verificar a evolução do conhecimento do usuário. Por exemplo, à medida que o usuário interage com o ambiente e responde corretamente a um teste sobre um determinado conceito, o FC relativo ao conhecimento do usuário é atualizado.

Neste caso, o teste que verifica o conhecimento do usuário sobre aquele conceito possui dois fatores de Confiança associados a ele. O fator FC_{know} indica a confiança do sistema sobre a pessoa conhecer o conceito, se ela selecionar a opção correta. Por outro lado, o $FC_{notKnow}$ indica a confiança do sistema de que a pessoa não conhece o conceito, se ela selecionar a errada. Ambos possuem a escala entre 0.0 a 1.0. A regra R6 determina o FC em função da resposta do usuário.

R6 – Determinação do FC_{know} e $FC_{notKnow}$ em função do acerto ou erro do teste Ti (i = 1, 2, ..., n)[5]

```
Se a resposta do teste T1 está correta então
      Usuário sabe o conteúdo com FC_know =  x
Caso contrário
      Usuário não sabe o conteúdo com FC_notknow =  x
```

Em seguida, o sistema utiliza uma das fórmulas da regra R4 para calcular o novo valor de FC_{user} para um conceito. Se o usuário selecionou a questão correta, o FC será FC_{know} e se ele respondeu o teste errado será $-FC_{notKow}$.

Toda vez que o usuário acertar uma questão, o modelo incrementa o FC e decrementa caso contrário. A cada cálculo do novo FC_{user} de um determinado conceito, aplica-se a regra R7 para verificar a necessidade de modificar o valor do *userLevel* do usuário sobre aquele conceito.

R7 – Atualização do *userLevel* do usuário

```
Se userLevel = "iniciante" e FC_new > w então
     userLevel = "intermediário"
Se userLevel = "intermediário" e FC_new > u então
     userLevel = "avançado"
```

Esses métodos estão armazenados em uma base de regras e são utilizados pelo Agente Pessoal (ver seção 4.6.2) que é o responsável pelo gerenciamento do MU. As informações do MU são inseridas em um SGBD e o acesso a esses dados é realizado através de consultas SQL. Esta estrutura permite armazenar o perfil de vários usuários mantendo a integridade das informações.

4.5 Modelo do Ambiente

O *Modelo do Ambiente* armazena a estrutura dos mundos virtuais utilizados pelo usuário. Esta estrutura permite gerar, recuperar e modificar um mundo utilizado pelo

[5] Para cada conteúdo podem existir diversos testes, denominados neste trabalho de T_1, T_2, ...T_n.

usuário. Toda vez que o usuário entra em um novo AV, este ambiente é associado ao usuário, com seu respectivo *userLevel*, e armazenado no Modelo do Ambiente.

A recuperação de um ambiente já visitado pelo usuário é realizada através de uma consulta ao Modelo do Ambiente, sendo recuperada a identificação do ambiente (*ID_Ambiente*) e o nome dos objetos pertencentes àquele ambiente. Esta ação é executada pelo Agente Ambiente (ver seção 4.6.3). Este mesmo procedimento pode ser realizado para se saber qual é o estado atual que está sendo utilizado pelo usuário. Isto permite ao Agente Atualizador (ver seção 4.6.4) gerar modificações no ambiente caso sejam necessárias.

Todas as modificações ocorridas no ambiente, sejam por inclusão ou remoção de novos elementos, ou por alteração das características dos objetos (por exemplo, nível de complexidade ou mudança de posição), são armazenadas no MA.

O Modelo do Ambiente utiliza a entidade Ambiente e os relacionamentos *Usuario_Ambiente* e *Ambiente_Objeto*, associando respectivamente cada usuário a um ambiente e cada ambiente aos objetos pertencentes ao ambiente gerado para o usuário. A cada novo ambiente gerado ou modificado, duas novas instâncias do relacionamento *Usuário-Ambiente* e *Ambiente-Objetos* são construídas. Tais relacionamentos garantem armazenar todos os ambientes gerados para cada usuário.

Esta estrutura é capaz de produzir relatórios de todas as modificações ocorridas durante uma sessão, bem como guardar o estado atual do ambiente ao final de uma sessão para recuperá-lo na próxima interação do usuário.

4.6 Atuação dos Agentes no VEPersonal

Os agentes são responsáveis pelo acompanhamento das ações do usuário e das modificações ocorridas no ambiente. Eles também são responsáveis pela definição de novos objetos a serem incluídos no mundo virtual e pela troca de mensagens com o usuário.

O emprego de agentes no processo de tomada de decisão permite construir sistemas eficientes e dinâmicos. Primeiro, porque a capacidade de comunicação entre agentes é facilitada por linguagens específicas. Segundo, porque a determinação de

objetivos, planos e ações para uma aplicação específica possibilita gerar um sistema de inferência eficiente.

Os agentes no VEPersonal são organizados em um sistema multi-agentes. Para tanto, os agentes foram organizados obedecendo aos princípios da arquitetura de um agente genérico em JADE, conforme descrito na Figura 3.6 da seção 3.4.3. Isto permite utilizar os recursos do ambiente JADE e agilizar o processo de comunicação e execução de tarefas.

Como apresentado anteriormente, os agentes que atuam no ambiente são: Agente Gerente, Agente Pessoal, Agente Ambiente e Agente Atualizador. A seguir, é descrita a arquitetura de cada agente com suas respectivas funções.

4.6.1 Agente Gerente

O Agente Gerente é responsável por coordenar as ações dos demais agentes para o cumprimento dos objetivos do sistema. Ele também é responsável por monitorar as ações do usuário através da comunicação com a interface.

A arquitetura interna do Agente Gerente é mostrada na Figura 4.10. As mensagens recebidas da interface, por meio de sensores (ver seção 5.4), possuem no seu conteúdo as ações realizadas pelo usuário. Uma base de comportamentos é usada para identificar o tipo de ação realizada e os comportamentos que devem ser executados para cada ação.

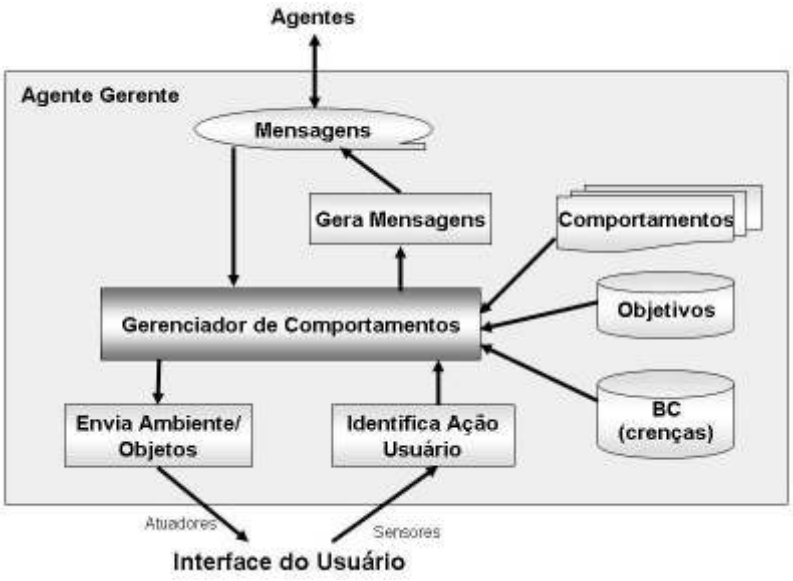


Figura 4.10 Arquitetura do Agente Gerente

O Agente Gerente identifica o tipo de ação do usuário e, em seguida, envia para o *Gerenciador de Comportamentos* que irá selecionar os comportamentos e gerar as mensagens para serem enviadas ao agente responsável. Por exemplo, se a ação do usuário corresponde à resposta a uma pergunta, o Agente Gerente deve enviar uma mensagem para o Agente Pessoal atualizar o perfil do usuário e, caso haja mudança no perfil, o Agente Gerente deve comunicar ao Agente Atualizador para que sejam providenciadas as devidas alterações no AV (geração de novos objetos) e retornadas ao Agente Gerente.

O Agente Gerente é um agente cognitivo porque, em função dos objetivos a alcançar (acompanhar a evolução do usuário, por exemplo), ele tem que elaborar um plano (alterar AV quando mudar perfil usuário) e determinar uma seqüência de ações para realizar a execução do plano[6] (solicitar perfil do usuário ao Agente Pessoal, solicitar ao Agente Ambiente o nome dos objetos do ambiente, solicitar Agente Atualizador o novo ambiente).

Quando o Agente Gerente recebe uma mensagem de um agente, o *Gerenciador de Comportamentos* identifica os comportamentos e dispara as ações necessárias (por exemplo, solicitar informações a outros agentes ou enviar ambiente para interface).

Caso o agente receba uma mensagem que determine alterações no ambiente, uma mensagem é gerada e enviada à interface solicitando a devida modificação do AV e inserindo no seu conteúdo os objetos 3D (recebidos do Agente Atualizador).

O Agente Gerente possui métodos que iniciam os comportamentos dos agentes responsáveis pelo tratamento de cada tipo de informação. O Gerenciador de Comportamentos controla as requisições enviadas aos agentes e gerencia as respectivas respostas. Essas requisições são realizadas através da troca de mensagens entre os agentes.

O comportamento do Agente Gerente é definido por um conjunto de regras armazenadas na sua base de Comportamentos. As principais ações do Agente Gerente que determinam o seu comportamento são:

- acompanhar a navegação do usuário e reconhecer as suas ações;
- identificar informações necessárias a atualização do MU;

[6] Os planos do Agente Gerente estão representados na sua Base de Conhecimentos (BC) e fazem parte do seu conjunto de crenças.

- identificar mudanças no ambiente para atualização do MA;
- monitorar escolhas anteriores do usuário (estilo de navegação, por ex.);
- solicitar ao *Agente Pessoal* o MU atual;
- solicitar ao *Agente Ambiente* o MA atual;
- solicitar ao Agente Atualizador as atualizações necessárias; e
- informar ao usuário que o seu perfil vai mudar e perguntar se ele aceita.

As mensagens geradas pelo Agente Gerente para o Agente Pessoal são:

- verificar se o login já existe e se a senha do usuário é correta;
- armazenar o cadastro e gerar o perfil de um novo usuário;
- solicitar o perfil de usuário já cadastrado;
- atualizar o perfil; e
- atualizar o cadastro.

As mensagens geradas pelo Agente Gerente para o Agente Ambiente são:

- verificar se o usuário já visitou um ambiente;
- solicitar ambiente visitado pelo usuário; e
- armazenar ambiente do usuário.

As mensagens geradas pelo Agente Gerente para o Agente Atualizador são:

- gerar ambiente para o usuário; e
- recuperar objetos para atualização do ambiente.

As mensagens trocadas entre o Agente Gerente e a Interface são:

- receber ações do usuário;
- enviar ambiente para o usuário; e
- enviar objetos 3D para atualização do ambiente.

Detalhes das mensagens geradas pelo Agente Gerente e do processo de comunicação com a Interface são apresentados no capítulo 5. Maiores informações sobre os protocolos de comunicação dos agentes do VEPersonal podem ser encontradas em Sales (2006).

4.6.2 Agente Pessoal

A principal função do *Agente Pessoal* é determinar o perfil do usuário em função das características do usuário (tais como preferências, intenções e interesses) e atualizar o perfil de acordo com as ações realizadas pelo usuário. Conforme discutido no capítulo 3, contar com um perfil de usuário atualizado é imprescindível para gerar ambientes adaptativos.

O Agente Pessoal é um agente reativo, pois só é acionado quando ocorre uma demanda do *Agente Gerente.* Além disso, o *Agente Pessoal* utiliza um sistema de inferência baseado em lógica para determinar o perfil, considerando as crenças que o sistema tem sobre o usuário.

Na Figura 4.11, é apresentada a arquitetura desse agente. As mensagens enviadas pelo Agente Gerente são analisadas pelo Gerenciador de Mensagens, responsável por identificar as solicitações do Agente Gerente e disparar os comportamentos necessários.

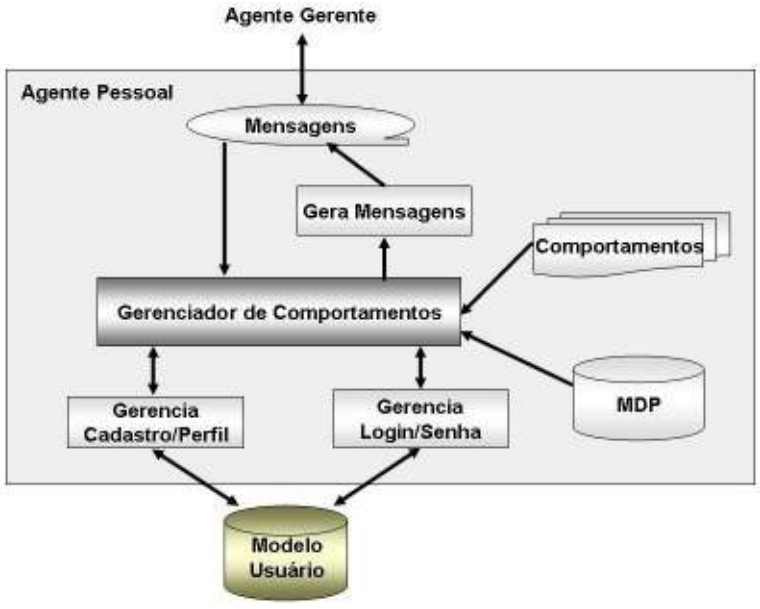


Figura 4.11 Arquitetura do Agente Pessoal

Na primeira vez que o usuário acessa o sistema, ele preenche um cadastro que é enviado ao Agente Pessoal. Neste momento, o agente calcula o perfil inicial do usuário com base nas informações fornecidas no cadastro. Este cálculo é realizado por uma base de regras, denominada *Métodos de Definição do Perfil (MDP)*, conforme discutido na seção 4.4. A vantagem desta estrutura é que os métodos podem ser alterados em função das técnicas empregadas na modelagem do usuário.

As ações do usuário, obtidas durante a sua interação com o ambiente, são enviadas do *Agente Gerente* para o *Agente Pessoal*, que também utiliza os Métodos de Definição de Perfil para atualizar o perfil do usuário e armazená-lo, juntamente com o cadastro, no Modelo do Usuário.

O Modelo do Usuário é armazenado em uma tabela com as informações pessoais do usuário, extraídas do cadastro, e com os dados gerados para o seu perfil (ver entidade *Usuario* da Figura 4.8).

Os comportamentos do Agente Pessoal estão definidos na sua base de Comportamentos. As principais ações do Agente Pessoal, que determinam o seu comportamento, são:

- validar *login* e senha do usuário;
- gerar o perfil do usuário a partir do cadastro;
- armazenar o cadastro e o perfil do usuário no MU;
- recuperar o perfil do usuário no MU caso o usuário já tenha um cadastro;
- atualizar o perfil e armazenar no MU; e
- atualizar o cadastro do usuário quando necessário.

As mensagens geradas pelo Agente Pessoal em resposta às perguntas do Agente Gerente são:

- informar se o *login* existe e se a senha do usuário é válida; e
- informar o perfil do usuário gerado.

4.6.3 Agente Ambiente

O Agente Ambiente é responsável por manter atualizadas as modificações ocorridas no ambiente do usuário e armazenar essas informações no Modelo do Ambiente. Este procedimento é importante porque possibilita a recuperação do ambiente, caso necessário. A recuperação pode ser útil caso ocorra uma queda de conexão com o sistema ou haja necessidade de carregar um ambiente já visitado pelo usuário.

Igualmente à atuação do Agente Pessoal, o Agente Ambiente é um agente reativo, pois só é acionado quando ocorre um estímulo, a partir da demanda do *Agente Gerente*.

A Figura 4.12 apresenta a arquitetura interna do Agente Ambiente. As mensagens recebidas do Agente Gerente são tratadas pelo Gerenciador de Comportamentos. Caso seja uma solicitação para recuperar um ambiente já visitado pelo usuário, uma consulta ao Modelo do Ambiente (MA) é realizada. Por outro lado, se o conteúdo da mensagem contiver o ambiente do usuário ou uma atualização, estes dados são armazenados no MA.

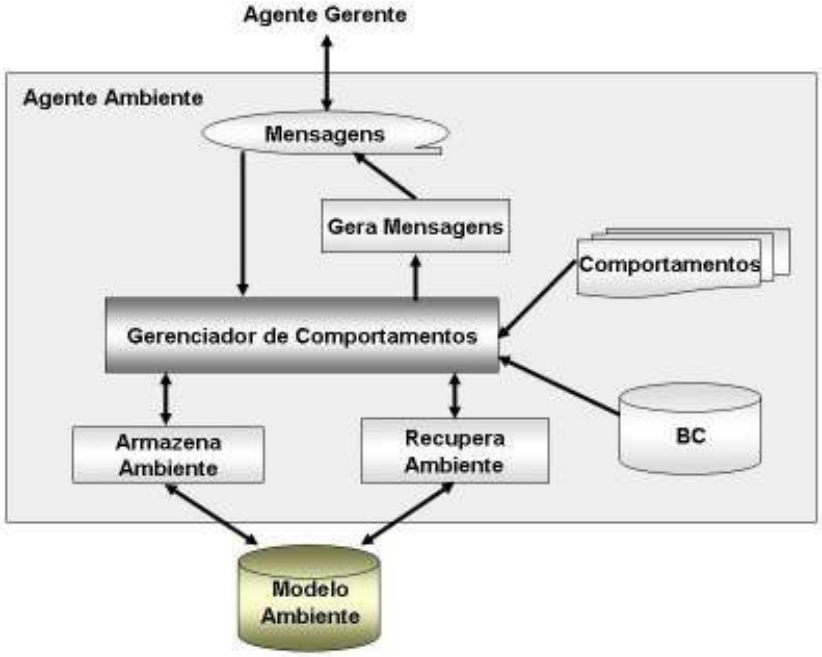


Figura 4.12 Arquitetura Interna do Agente Ambiente

O Modelo do Ambiente é estruturado em uma tabela composta de: identificador do ambiente gerado para o usuário (*ID_Ambiente*) e os nomes dos objetos (*ID_Objeto*) que pertencem ao ambiente (ver relacionamento Ambiente_Objetos da Figura 4.8).

O MA é modificado sempre que o *Agente Atualizador* gerar um novo mundo para o usuário ou forem definidos novos objetos para serem inseridos no ambiente atual.

As atualizações são comunicadas ao Agente Ambiente, informando a lista dos nomes dos objetos pertencentes ao ambiente. Com estes nomes, o Gerenciador de Comportamentos dispara o comportamento responsável por atualizar a lista de objetos do ambiente que está armazenada no MA.

Sendo assim, toda vez que o Agente Gerente enviar uma mensagem solicitando informações atualizadas do ambiente onde o usuário se encontra, o Agente Ambiente fornece tais informações que estão armazenadas no MA. O gerenciamento da inserção ou atualização dos ambientes no MA, bem como da recuperação de tais ambientes quando necessário, é controlado pela BC do Agente Ambiente.

As ações a serem realizadas pelo *Agente Ambiente* são:

- armazenar as informações do ambiente do usuário no *Modelo do Ambiente*;
- atualizar o ambiente do usuário no *Modelo do Ambiente*; e
- informar, quando solicitado pelo *Agente Gerente*, o ambiente atual do usuário.

As mensagens geradas pelo Agente Ambiente para o Agente Gerente são:

- confirmar ou negar que o ambiente foi visitado pelo usuário; e
- se usuário já visitou o ambiente, informar os dados do ambiente (identificador do ambiente e a lista de objetos que compõem o ambiente).

4.6.4 Agente Atualizador

O *Agente Atualizador* é o responsável por determinar a atualização do mundo do usuário e obter novos objetos do SGBD para serem incluídos no ambiente virtual. É um agente reativo porque ele só é acionado quando o Agente Gerente necessita gerar um

ambiente para o usuário ou atualizar o já existente. A arquitetura do Agente Atualizador é exibida na Figura 4.13.

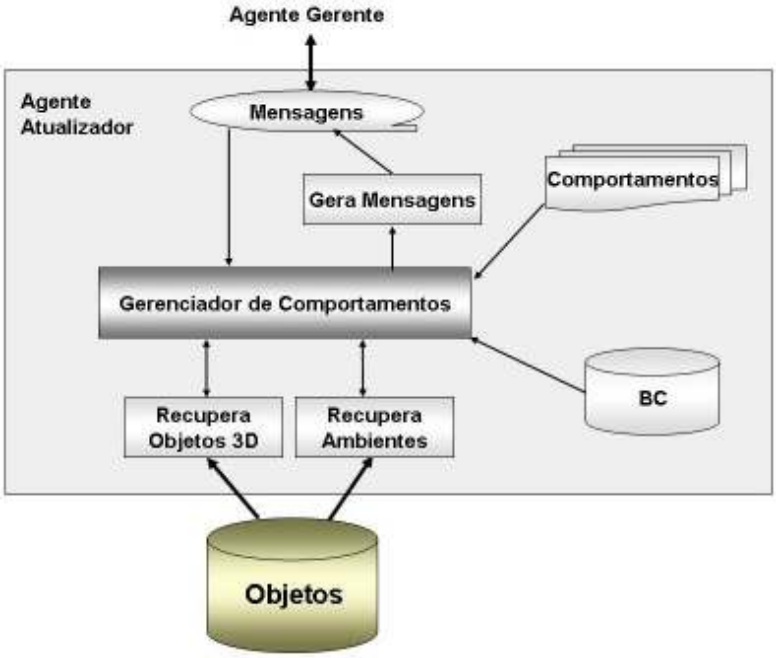


Figura 4.13 Arquitetura do Agente Atualizador

Um dos objetivos do Agente Atualizador é gerar um ambiente inicial para o usuário. Para tanto, ele recebe a identificação do usuário (*ID_Usuario*), o seu perfil e o nome do ambiente que deve ser gerado. Em seguida, o Agente Atualizador recupera os objetos que irão compor o ambiente virtual, armazenados no SGBD, e os envia para o Agente Gerente. A recuperação dos objetos obedece a uma consulta ao SGBD utilizando as entidades *Objetos* e *Ambiente* e o relacionamento *Ambiente_Objetos* da Figura 4.8.

A atualização do ambiente que está sendo utilizado pelo usuário é outro objetivo que o Agente Atualizador deve realizar. Isto ocorre quando o perfil do usuário foi modificado e, conseqüentemente, objetos devem ser adicionados ou retirados do ambiente virtual. Neste caso, o agente precisa conhecer o conteúdo do ambiente atual que está sendo utilizado pelo usuário. Este conteúdo, assim como o perfil, é recebido através de uma mensagem enviada pelo Agente Gerente.

As regras que determinam a atualização do usuário estão armazenadas na BC do Agente Atualizador. Esta atualização deve seguir uma seqüência de ações determinadas pelo Agente Atualizador. As ações a serem realizadas são:

- identificar objetos necessários à inclusão, retirada ou modificação no ambiente;
- gerar consultas ao SGBD para recuperar os objetos 3D;
- enviar objetos, recuperados do SGBD, para o *Agente Gerente*; e
- enviar informações da atualização do ambiente ao *Agente Gerente*.

4.7 Interface de Comunicação Cliente/Servidor

O VEPersonal utiliza um ambiente Web para a comunicação com o usuário que está conectado remotamente. Uma interface de comunicação entre o usuário e o sistema foi desenvolvida para realizar a sincronização da informação, tendo em vista ser um ambiente dinâmico.

A função da interface é comunicar as ações do usuário ao *Agente Gerente*, além de receber os objetos para atualização do mundo virtual. Essa atualização pode ser através de um ambiente X3D completo, inserção de objetos (*tags* X3D) ou remoção de objetos (por nome). A interface deve disponibilizar meios para que o *Agente Gerente* possa realizar essas operações.

A visualização de mundos virtuais pelo usuário é realizada pelo *browser* Xj3D [Xj3D, 2007]. Este *browser* é *open source*, desenvolvido em Java pela Web3D Consortium, e permite visualizar ambientes X3D e VRML. Aplicações usando Xj3D têm sido desenvolvidas para simulação, treinamento e ensino, bem como para visualização médica, jogos e projetos envolvendo sensores de visualização [Matsuba, 2006].

Além desse *browser*, foi utilizado a API *Scene Access Interface* (SAI) [SAI, 2007] para fazer a comunicação entre os ambientes X3D e Java. Esta API permite a inserção, remoção e modificação dos objetos no *browser* Xj3D em tempo real.

É possível realizar um acesso externo ao ambiente virtual através de uma classe Java que chama uma instância do *browser* e carrega um arquivo X3D. Em seguida, SAI é utilizada para receber as informações do *browser* ou efetuar alguma ação na cena (por exemplo, inserir ou remover objetos) [Martins, 2007].

A integração desses recursos no ambiente Web utilizou a tecnologia *Java Web Start* (JWS) [JWS, 2007] com o objetivo de possibilitar a execução de aplicativos Web,

controlando os serviços e os protocolos de comunicação. O *Java Web Start* é iniciado automaticamente quando é feito o primeiro *download* da aplicação do VEPersonal.

O JWS armazena o aplicativo localmente, assim, todas as inicializações subseqüentes são quase instantâneas, pois todos os recursos necessários já estão disponíveis.

No lado do cliente, o usuário acessa o sistema via browser e pode realizar funcionalidades como cadastro e autenticação. Quando ele solicita a geração do mundo 3D, uma aplicação Java é iniciada para monitorar as ações do usuário no ambiente e informar ao servidor a execução de uma ação (como clique do mouse ou proximidade a objetos). Esta aplicação também é responsável por receber o mundo a ser carregado ou os objetos a serem incluídos e/ou excluídos no mundo virtual (Figura 4.14).

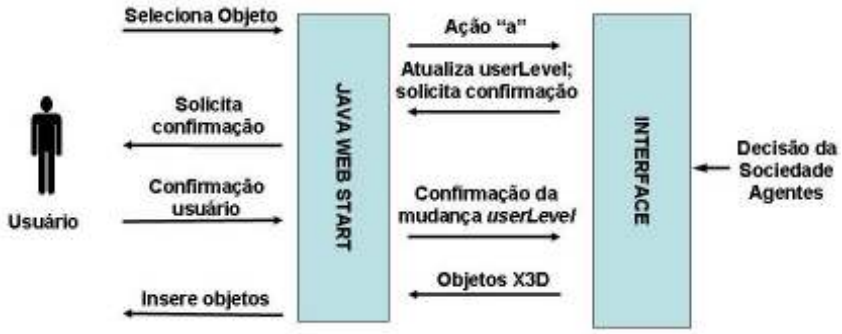


Figura 4.14 Interface cliente-servidor do VEPersonal utilizando Java Web Start

No lado do servidor, quando uma ação do usuário é recebida, o *Agente Gerente* se encarrega de determinar quais devem ser as ações a executar. Estas ações são transformadas em mensagens do protocolo e enviadas aos demais agentes. A sociedade de agentes toma as decisões necessárias e retorna à interface, a atualização do mundo virtual, caso necessário.

Os Casos de Uso apresentados na Figura 4.15 representam as ações realizadas pelo *Agente Gerente*, pela *Interface* do VEPersonal e pelo *Usuário* e são detalhados a seguir.

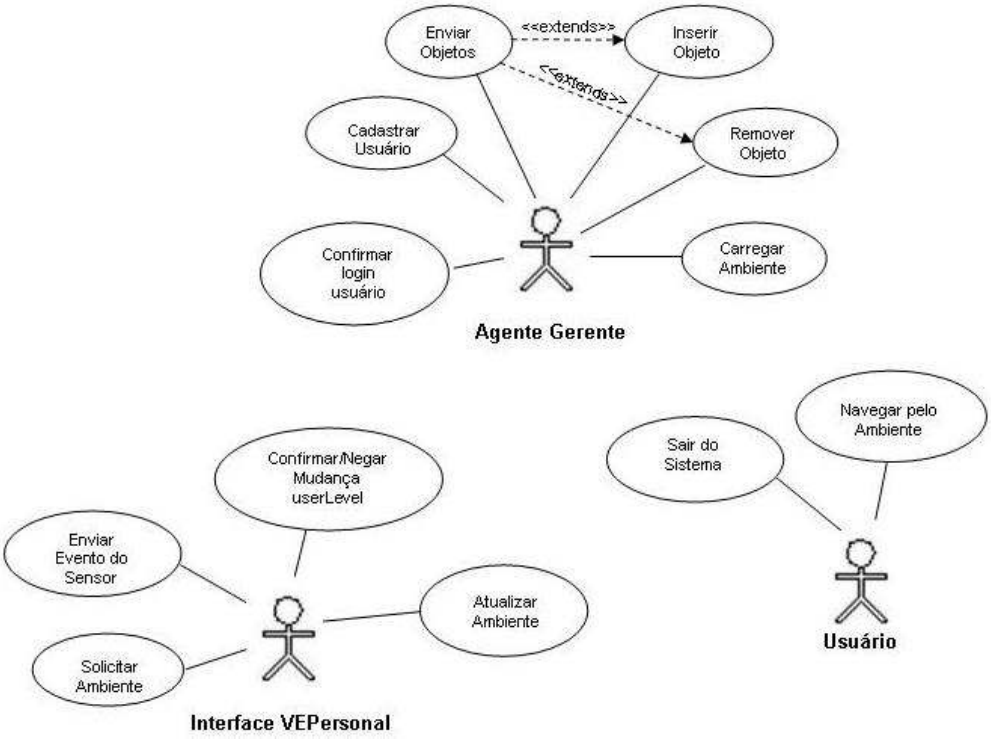


Figura 4.15 Diagrama de Casos de Uso da comunicação com o usuário

A comunicação do Agente Gerente com a Interface é realizada pelas seguintes ações:

- Cadastrar Usuário – o Agente Gerente recebe da Interface o cadastro do usuário que está se conectando pela primeira vez no sistema. Retorna o *ID_Usuario* (código de identificação) e o *userLevel* deste usuário;
- Confirmar Login Usuário – responsável por gerar o ambiente para um usuário já cadastrado, após este ter realizado a autenticação no sistema. Fornece ao sistema, o *ID_Usuario* e o *userLevel*;
- Carregar Ambiente – esse caso de uso é chamado após a confirmação do *login* do usuário ou do cadastramento de um novo usuário. O Agente Gerente envia para a Interface o ambiente a ser apresentado ao usuário. Este caso de uso também é executado quando o usuário muda de ambiente;
- Enviar Objetos – é um requisito fundamental para a adaptação em tempo real. Envia um conjunto de objetos à Interface, associados a uma ação para cada um. Essa ação pode ser inserção, remoção ou modificação desses objetos no ambiente. Este caso de

uso permite que se faça uma série de alterações no ambiente com apenas um comando;

- Inserir Objeto – este caso de uso é responsável pela inserção de objetos na cena;
- Remover Objeto – é responsável por remover objetos da cena. Esta remoção é realizada fornecendo o nome do objeto a ser removido; e
- Modificar Objeto – para que um objeto seja modificado, remove-se do ambiente o objeto e insere-se o objeto atualizado.

A seguir, serão discutidos os casos de uso relativos às chamadas da Interface ao Gerente:

- Solicitar Ambiente – esse caso de uso é chamado após a confirmação de um login. Deverá solicitar ao Agente Gerente o ambiente correspondente ao usuário recém-admitido pelo sistema, que será então enviado por meio do caso de uso Carregar Ambiente;
- Enviar Evento de Sensor – sempre que o usuário interagir com algum tipo de sensor no ambiente (sensores de toque e de proximidade), a interface deverá comunicar ao Agente Gerente o *id* do usuário e o nome do sensor ativado;
- Confirmar/Negar Mudança de *userLevel* – este caso de uso prevê a consulta ao usuário sobre a mudança de *userLevel* (e conseqüente atualização do ambiente virtual); e
- Atualizar ambiente – um dos principais requisitos da Interface é a inserção e remoção de objetos sem que seja necessário recarregar toda a cena (requisito fundamental da adaptação dos ambientes em tempo real). Portanto, a Interface deverá fornecer esse serviço ao Agente Gerente, que será realizado pelo caso de uso *Enviar Objetos*.

Os Casos de Uso relacionados ao usuário são navegar no ambiente e sair do sistema. Estas ações requerem apenas a interação do usuário através de clique de mouse nos objetos e resposta às perguntas realizadas (também realizada pela interação com

objetos 3D), além de deslocamento pelo ambiente. A saída do sistema é realizada pelo deslocamento do usuário em direção à porta indicada por *SAÍDA*.

4.8 Considerações Finais

Ambientes Virtuais Adaptativos têm sido propostos com o objetivo de personalizar o ambiente em função das necessidades e do conhecimento do usuário, bem como determinar níveis de acesso a ambientes específicos. Esta adaptação visa aumentar a satisfação e produtividade do usuário em ambientes tridimensionais.

Neste capítulo, foi proposta uma arquitetura para a implementação de Ambientes Virtuais Adaptativos, atendendo às premissas discutidas no início deste capítulo: (i) uma aplicação cliente-servidor contribui para a performance da interface de comunicação com o usuário via Web; (ii) o SGBD com suporte a XML armazena tanto os dados do usuário e do estado atual do ambiente (que possuem estrutura relacional), bem como os objetos tridimensionais (que possuem estrutura XML). Neste caso, uma única interface com o SGBD é utilizada, facilitando o gerenciamento e a recuperação desses dados; (iii) uma sociedade de agentes é responsável pelo acesso às informações armazenadas no SGBD e pelo processo de análise e tomada de decisão. Por ser um ambiente multi-agentes, a distribuição de tarefas agiliza esse processo e realiza, em tempo real, as atualizações necessárias no ambiente do usuário.

A implementação da arquitetura inclui alguns requisitos necessários à adaptativdade, não encontrados nos sistemas descritos na literatura. Tais requisitos são: a apresentação de conteúdos em níveis de complexidade diferentes; reuso de objetos; e adaptação em tempo real.

A geração de ambientes virtuais com diversos níveis de complexidade foi realizada com a introdução do atributo *userLevel*, que representa o estereótipo do usuário. Isto permite construir ambientes virtuais em que é possível realizar uma projeção gradativa de informações à medida que é alterado o perfil do usuário. Uma das vantagens, neste caso, é a possibilidade de se ter um mesmo modelo do ambiente para níveis de usuários diferentes, além de facilitar a atualização do ambiente virtual com a mudança do perfil do usuário.

O reuso de um mesmo objeto em vários AV pôde ser realizado a partir do armazenamento dos objetos 3D em um SGBD com suporte a XML. O processo de construção do AV é realizado através da recuperação apenas dos objetos necessários ao ambiente. Desta forma, novos ambientes podem utilizar objetos que foram inicialmente concebidos para outros AV.

O SGBD PostgreSQL foi escolhido para ser utilizado no VEPersonal porque satisfaz tanto as necessidades de armazenamento e recuperação de objetos 3D quanto de dados relacionais. Além disso, a integração com os agentes é facilitada com o uso da interface JDBC. Assim, cada agente pode acessar o SGBD de acordo com o tipo de dados que ele deseja armazenar ou recuperar.

O Modelo do Usuário permite armazenar o perfil dos usuários cadastrados no VEPersonal, facilitando a recuperação desses dados para geração de um ambiente virtual personalizado. Esta estrutura possibilita o gerenciamento de vários ambientes simultaneamente, tendo em vista o armazenamento dos dados de cada usuário no SGBD com uma identificação única. A manutenção do MU, representada por regras de geração e atualização do perfil do usuário são realizadas por agentes, facilitando o acompanhamento e a monitoração das ações do usuário e de sua evolução cognitiva.

O Modelo do Ambiente foi incluído na arquitetura devido à necessidade de armazenar o estado atual do AV durante a conexão do usuário e realizar um histórico das suas modificações. Isto permite recuperar o AV caso ocorra uma interrupção de comunicação ou queda do sistema. Armazenar o estado do mundo virtual também possibilita recuperar a situação desse mundo quando da última visita do usuário, ou seja, em uma nova conexão, o usuário pode recomeçar de onde havia parado.

A utilização de agentes para gerenciar as ações de um ambiente tridimensional adaptativo é uma proposta inovadora no sentido de colocar as decisões do sistema em entidades autônomas e portadoras de inteligência. Os agentes têm capacidade de avaliar o ambiente e tomar decisões, o que possibilita construir um sistema capaz de acompanhar as ações do usuário (através de sensores), inferir sobre o usuário e as condições do ambiente, e gerar a adaptação necessária (através de atuadores). As regras de personalização e adaptatividade, necessárias para o ambiente proposto, foram inseridas na base de conhecimento dos agentes, gerando maior autonomia às suas ações.

O uso de um sistema multi-agentes permite a distribuição de tarefas e conseqüentemente a modularização das atividades de consulta aos dados, impedindo que mais de um agente acesse os mesmos dados. O ambiente JADE, por sua vez, possibilitou a comunicação dos agentes de forma padronizada e autônoma, facilitando a troca de informações entre eles.

As interfaces para ambientes 3D têm sido aprimoradas nos últimos anos. Novos recursos surgiram com a padronização da linguagem X3D, como por exemplo, a criação de *browsers* específicos para tais ambientes (Xj3D) e uma interface de programação (SAI) para inserção e remoção de objetos nesse *browser*. Tais interfaces geram condições favoráveis ao gerenciamento de AV Adaptativos em tempo real.

Desta forma, a interface da arquitetura proposta utiliza tais recursos com o objetivo de capturar as ações do usuário (aproximação ou seleção de objetos) e modificar o ambiente (inserção de novos objetos, por exemplo). Considerando que esta interface é usada em ambiente Web, é necessário adicionar um novo recurso para o seu funcionamento, o *Java Web Start* (JWS). Esta tecnologia permite que aplicações possam ser iniciadas diretamente da Internet utilizando o browser Xj3D[7].

No próximo capítulo, são apresentados os resultados obtidos na aplicação do VEPersonal em um ambiente de ensino de Física. Uma avaliação do desempenho do sistema é realizada em função das tecnologias utilizadas.

[7] O browser Xj3D foi implementado em Java e pode ser integrado com outras aplicações Java, inclusive o Java Web Start (http://www.3d-test.com/interviews/xj3d_1.htm).

5. VEPersonal: Metodologia de Desenvolvimento e Implementação

O VEPersonal foi concebido para ser uma ferramenta independente de plataforma e poder utilizar recursos computacionais de baixo custo para executar aplicações de RV.

Para atingir tais objetivos, foram utilizadas tecnologias *open source*, consolidadas no mercado, que pudessem ser executadas em diversos sistemas operacionais como Linux [Linux, 2007], Windows [Windows, 2007], Solaris [Solaris, 2007] ou qualquer plataforma que possua uma versão JVM (*Java Virtual Machine*) [JVM, 2007]. Também optou-se por um AV não-imersivo por não necessitar de acessórios especiais como luvas, óculos ou capacete. Esta solução considerou também o acesso de usuários via Web conectados apenas a um microcomputador *desktop* com monitor 2D.

Apesar de que, *a priori*, qualquer computador pode ser utilizado para acessar o VEPersonal, recomenda-se uma configuração mínima de um processador de 1 GHz, memória RAM de 512 Mb, placa de vídeo colorida SVGA e conexão com a Internet de no mínimo 612 kbps. Esta especificação é sugerida apenas para que a aplicação não tenha perda de desempenho e não comprometa o tempo de resposta para visualização e interação com o ambiente virtual.

O processo de desenvolvimento do VEPersonal utilizou a metodologia *Extreme Programming*, também conhecida como XP [XP, 2007]. Esta metodologia enfatiza o uso intenso de práticas funcionais, tais como revisões de código, testes, simplicidade e ciclos curtos e iterativos [Beck, 2000]. A utilização da metodologia XP permitiu desenvolver com maior eficiência a integração dos módulos do VEPersonal (interface, sistema multi-agentes e SGBD).

Diversos padrões de projeto (*Design Patterns*) foram utilizados com o objetivo de padronizar a integração do software e torná-lo independente das tecnologias empregadas, permitindo a possibilidade de reuso, manutenção e facilidade no seu desenvolvimento. Dentre os padrões de projeto utilizados [Gama et al., 2002],

destacam-se os padrões GoF (*Gang of Four*): *Singleton* (no reaproveitamento da conexão no SGBD), *Adapter*, *Façade* (na conexão cliente/servidor), *Observer* (para criar o *Listener* do browser Xj3D); os padrões JEE (*Java EE*): *Data Access Object* – DAO (no acesso ao SGBD), *Filter* (verifica login/senha, diminuindo o *overhead*); e os padrões GRASP (*General Responsibility Assignment Software Patterns*): *Information Expert* (praticas de orientação de projeto), *Low Coupling* e *High Cohesion* (utilizados na comunicação cliente/servidor).

As tecnologias utilizadas pelo VEPersonal têm como base de desenvolvimento a linguagem Java. Os agentes foram concebidos no ambiente JADE que, como discutido na seção 3.4.3, é um *framework* para desenvolvimento de sistemas multi-agentes baseado em Java. O *browser* Xj3D utilizado para visualização de ambientes 3D (ver seção 4.7) também foi desenvolvido em Java o que facilitou a sua integração à interface. Para comunicação via Web, foi utilizado o *Java Web Start*, cuja tecnologia permitiu a integração do *browser* com o servidor.

O SGBD PostgreSQL, escolhido para o armazenamento e gerenciamento dos objetos 3D (discutido nas seções 3.2.2 e 4.3), possui interface JDBC que permite a integração de aplicações Java ao SGBD, facilitando o acesso dos agentes.

Este capítulo descreve as implementações realizadas no VEPersonal. Inicialmente, a seção 5.1 apresenta o ambiente de estudo utilizado para a realização dos testes e suas funcionalidades. Nas seções seguintes, utilizando um exemplo do ambiente de testes, são detalhados o armazenamento e gerenciamento de objetos no SGBD (seção 5.2), o comportamento dos agentes e a comunicação entre eles (seção 5.3), a estrutura da interface de comunicação cliente/servidor e as soluções encontradas para a integração dessas ferramentas (seção 5.4).

Por fim, uma avaliação dos resultados obtidos é realizada, fazendo-se uma análise do tempo de resposta do sistema para os acessos do usuário e para as modificações ocorridas no ambiente (seção 5.5).

5.1 O Ambiente de Estudo

A infra-estrutura do VEPersonal foi utilizada para a construção do ambiente de estudo que serviu de experimentação e validação do trabalho desenvolvido. Este

ambiente utilizou a plataforma Windows XP [WindowsXP, 2007] e foi concebido com o objetivo de ensinar Física a alunos do 2º grau. Neste contexto, os alunos podem interagir com experimentos, acessar informações sobre os seus conteúdos e responder a testes sobre os conteúdos estudados. Cada teste é específico para o nível de conhecimento do usuário.

O acesso ao VEPersonal é realizado a partir de uma página HTML, construída com informações sobre o projeto. Utilizou-se o *framework* Struts 2 [Struts2, 2007], desenvolvido em Java, que possui uma camada de controle baseada nas tecnologias Java Servlets, JavaBeans, ResourceBundles e XML. Este *framework* favoreceu o desenvolvimento da aplicação seguindo o paradigma *Model View Controller*[8], que fornece um componente Controller e se integra a outras tecnologias para oferecer suporte aos componentes Model (com JDBC) e View (com JSP e XSLT).

Na Figura 5.1, é apresentada a página inicial do VEPersonal. Nesta página, os usuários têm as seguintes opções: obter informações sobre o VEPersonal; realizar o cadastramento no sistema, informando os dados pessoais, as preferências, interesses e nível de conhecimento; e realizar *login*, que permite o acesso à aplicação e à interação com os AV desenvolvidos no projeto.

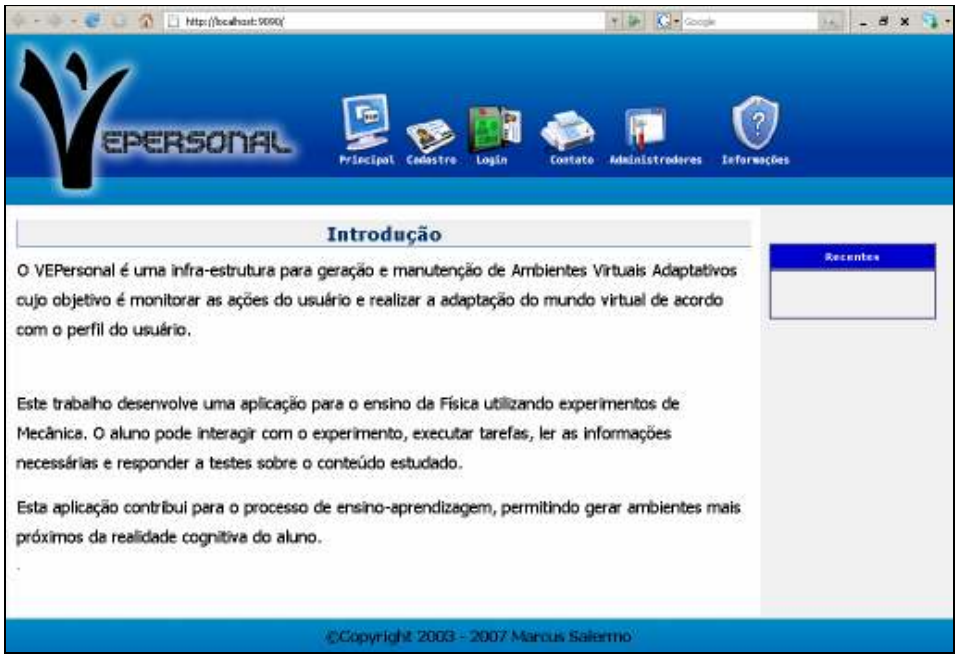


Figura 5.1 Página inicial do VEPersonal

[8] *Model View Controller* é um padrão de arquitetura de aplicações que visa separar a lógica de aplicação (*Model*) da interface do usuário (*View*) e do fluxo da aplicação (*Controller*). Permite que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces.

Na Figura 5.2, é apresentada a página de autenticação do usuário no VEPersonal, na qual o usuário pode fazer a conexão com o sistema. Nesta página, o usuário deve informar *login* e senha. O controle de acesso ao sistema é realizado através da inclusão, exclusão e edição dos dados do usuário em relação ao *login* e senha diretamente no SGBD, agilizando a validação do *login*.

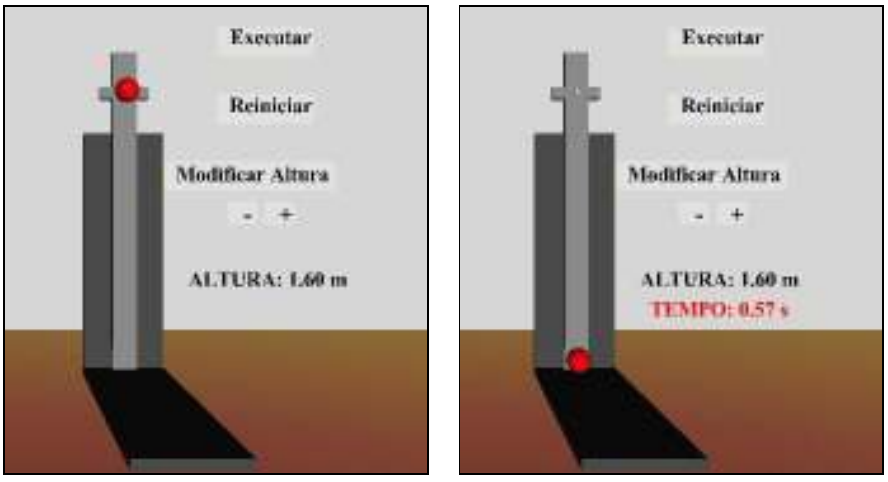
Figura 5.2 Página de autenticação do usuário no sistema

Após a confirmação do login e senha, o ambiente de estudo composto por um "*hall de entrada*" é apresentado. Neste ambiente, o estudante (usuário) visualiza três portas. Cada porta dá acesso a uma sala contendo experimentos mecânicos de Cinemática, Dinâmica e Mecânica Estática (Figura 5.3).

O estudante é orientado para interagir com os experimentos, executando um conjunto de tarefas propostas pelo sistema. Ele pode navegar pelas salas e sair delas de acordo com sua conveniência. O sistema é capaz de armazenar o estado corrente do ambiente (no Modelo do Ambiente) e as informações atuais do usuário (no Modelo do Usuário). Tais informações serão recuperadas quando o estudante retornar ao ambiente.

Figura 5.3 Hall de Entrada da aplicação Experimentos de Física visto pelo usuário

Na Figura 5.4, é apresentada como exemplo, a interação do estudante com o experimento Queda Livre. O objetivo deste experimento é estudar a aceleração da gravidade. Ele é composto por uma estrutura chamada *KitVertical* e uma esfera posicionada a 1,60 m de altura, além de botões que permitem interagir com a aplicação. O estudante executa o experimento selecionando o botão `EXECUTAR` (Figura 5.4(a)). Em seguida, a esfera realiza um movimento descendente, sob o efeito da ação da gravidade, até chegar à base. Neste momento, o tempo realizado durante a queda é apresentado ao estudante (Figura 5.4(b)). O estudante também pode alterar a altura da esfera e, após a execução do experimento novamente, o tempo de queda correspondente é apresentado.



(a) Início do experimento (b) Experimento após execução

Figura 5.4 Experimento Queda Livre

Para cada experimento, o estudante pode consultar a teoria do conteúdo e executar interações. Em função das interações realizadas pelo usuário, um teste de avaliação é apresentado. Este teste é específico para o nível de conhecimento do estudante (de acordo com o perfil armazenado no Modelo do Usuário).

O teste para um estudante iniciante, apresentado na Figura 5.5, é composto de uma pergunta e três opções de resposta. A inserção do teste no ambiente é determinada por agentes que recebem informações da interface sobre as ações do usuário. As regras de negócio dos agentes, que determinam o que modificar e quando isso deve ocorrer, são apresentadas na seção 5.2.

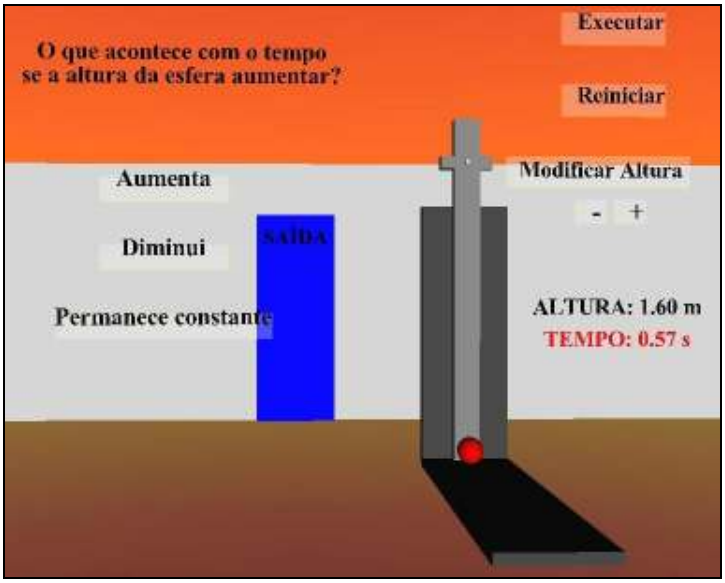


Figura 5.5 Teste para iniciantes

Caso o usuário seja um estudante experiente, uma pergunta mais elaborada (com grau de dificuldade maior) é apresentada (Figura 5.6).

Em toda sala que o usuário estiver interagindo com um experimento, sempre haverá uma porta (SAÍDA) por onde ele poderá sair a qualquer momento e encerrar a sessão.

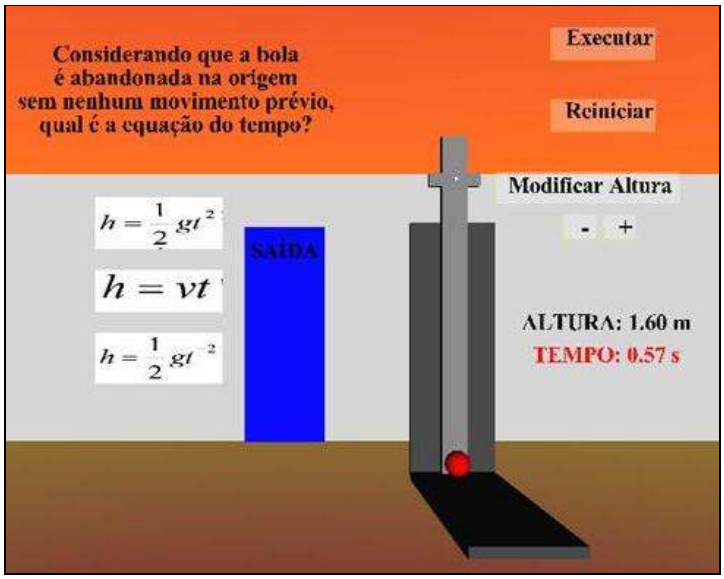


Figura 5.6 Teste para estudantes experientes

Outra vantagem para o uso do perfil do usuário nesta aplicação é a determinação de acesso a outros ambientes. Por exemplo, o acesso a uma outra sala, que contém outros experimentos com o mesmo conceito, só poderá ocorrer para estudantes que já atingiram um nível de conhecimento considerado adequado. A Figura 5.7 descreve a visão de um estudante iniciante que não tem permissão de entrar em outra sala (porque ele provavelmente não tem o nível de conhecimento necessário).

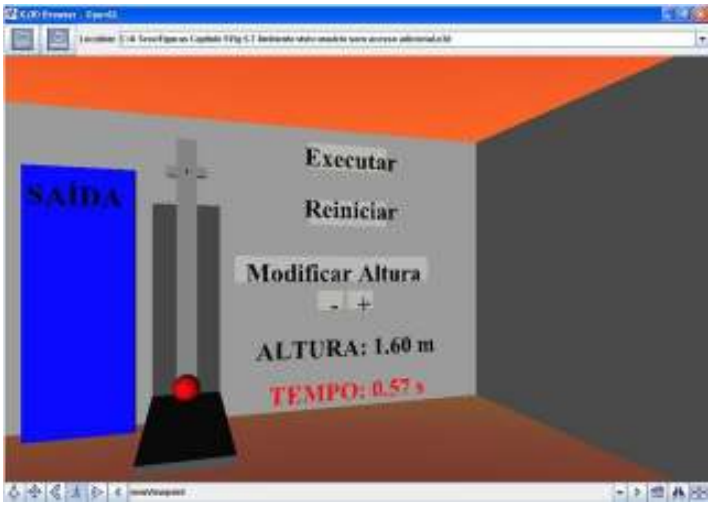


Figura 5.7. Ambiente visto por usuário sem acesso adicional

Por outro lado, a Figura 5.8 mostra uma porta vermelha à direita que permite ao estudante, entrar (ou seja, ele tem conhecimento apropriado para isso). Este exemplo é o

resultado de uma adaptação das características do usuário que estão armazenadas no Modelo do Usuário.

Figura 5.8 Ambiente visto por usuário com acesso adicional

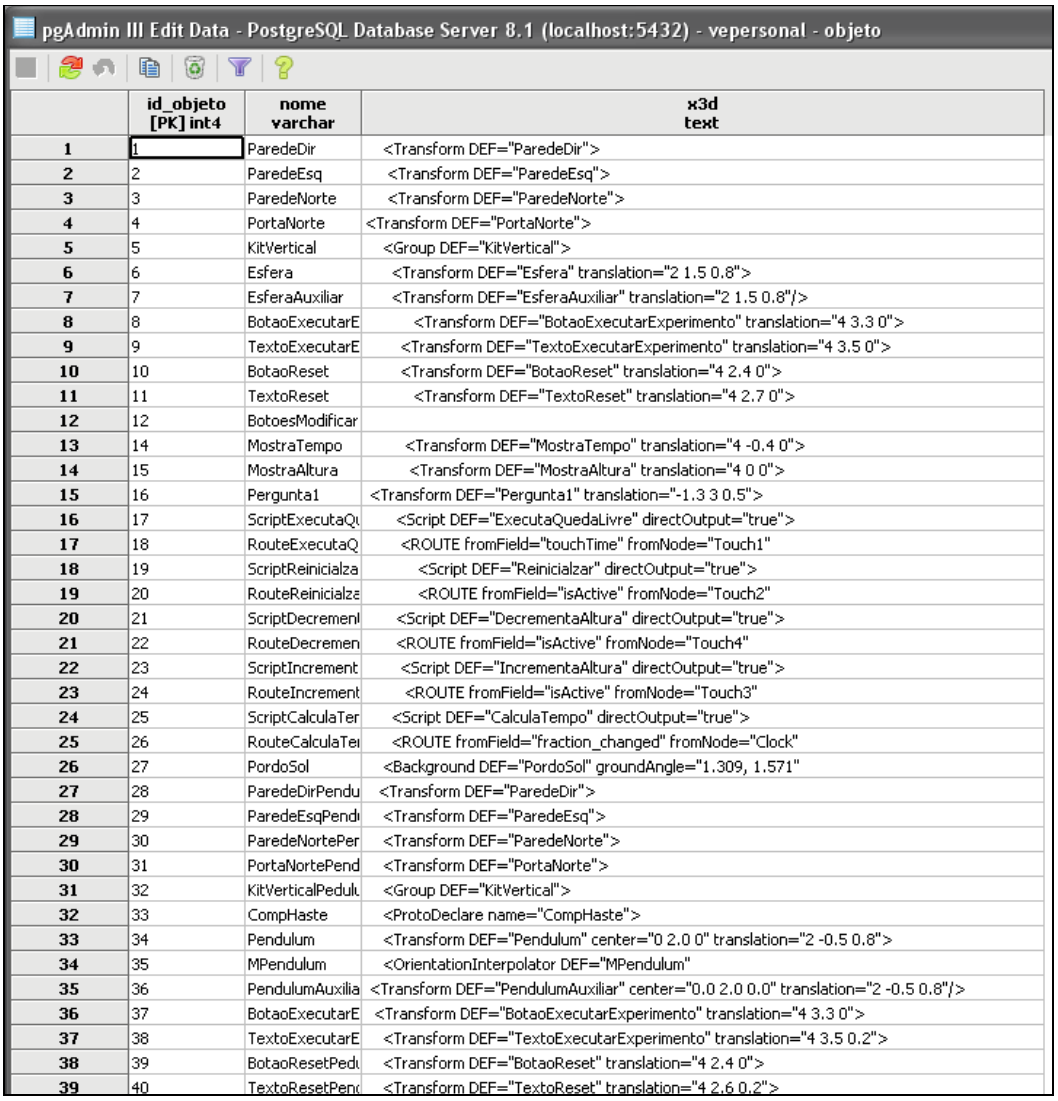
5.2 Armazenamento e Gerenciamento de Objetos

O VEPersonal utiliza o SGBD PostgreSQL, versão 8.1, para armazenar os objetos X3D e todas as informações referentes a estrutura dos ambientes virtuais gerados para o usuário.

As conexões com o banco de dados são controladas pelo *framework* Proxool [Proxool, 2007], que permite criar e gerenciar um *pool* de conexões, monitorando a performance das conexões e acompanhando os respectivos eventos. O JMetter (ver seção 3.2.3) pode, neste caso, ser utilizado para acompanhar o fluxo de conexões.

Os objetos X3D, que possuem estrutura XML, são armazenados em campos *text*, podendo ser acessados por consultas XPath. Uma consulta XPath é capaz de selecionar os objetos com os seus respectivos níveis, representados por *tags* identificadas pelo *userLevel* (ver seção 4.2). As tabelas utilizadas nos exemplos desta seção são determinadas pelo modelo de dados do sistema de banco de dados descrito na seção 4.3 do capítulo 4.

A Figura 5.9 apresenta os objetos 3D armazenados no SGBD. Nesta tabela (Tabela *Objetos*), todos os objetos dos ambientes virtuais são armazenados. Cada objeto é composto por um identificador (*id_objeto*), um *nome* e o código *X3D*.

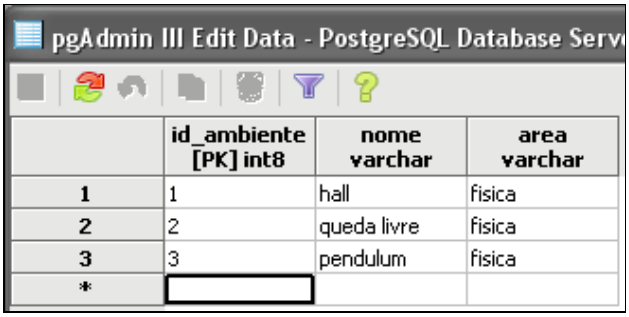


pgAdmin III Edit Data - PostgreSQL Database Server 8.1 (localhost:5432) - vepersonal - objeto

	id_objeto [PK] int4	nome varchar	x3d text
1	1	ParedeDir	<Transform DEF="ParedeDir">
2	2	ParedeEsq	<Transform DEF="ParedeEsq">
3	3	ParedeNorte	<Transform DEF="ParedeNorte">
4	4	PortaNorte	<Transform DEF="PortaNorte">
5	5	KitVertical	<Group DEF="KitVertical">
6	6	Esfera	<Transform DEF="Esfera" translation="2 1.5 0.8">
7	7	EsferaAuxiliar	<Transform DEF="EsferaAuxiliar" translation="2 1.5 0.8"/>
8	8	BotaoExecutarE	<Transform DEF="BotaoExecutarExperimento" translation="4 3.3 0">
9	9	TextoExecutarE	<Transform DEF="TextoExecutarExperimento" translation="4 3.5 0">
10	10	BotaoReset	<Transform DEF="BotaoReset" translation="4 2.4 0">
11	11	TextoReset	<Transform DEF="TextoReset" translation="4 2.7 0">
12	12	BotoesModificar	
13	14	MostraTempo	<Transform DEF="MostraTempo" translation="4 -0.4 0">
14	15	MostraAltura	<Transform DEF="MostraAltura" translation="4 0 0">
15	16	Pergunta1	<Transform DEF="Pergunta1" translation="-1.3 3 0.5">
16	17	ScriptExecutaQu	<Script DEF="ExecutaQuedaLivre" directOutput="true">
17	18	RouteExecutaQ	<ROUTE fromField="touchTime" fromNode="Touch1"
18	19	ScriptReinicialza	<Script DEF="Reinicialzar" directOutput="true">
19	20	RouteReinicialza	<ROUTE fromField="isActive" fromNode="Touch2"
20	21	ScriptDecrementl	<Script DEF="DecrementaAltura" directOutput="true">
21	22	RouteDecremen	<ROUTE fromField="isActive" fromNode="Touch4"
22	23	ScriptIncrement	<Script DEF="IncrementaAltura" directOutput="true">
23	24	RouteIncrement	<ROUTE fromField="isActive" fromNode="Touch3"
24	25	ScriptCalculaTer	<Script DEF="CalculaTempo" directOutput="true">
25	26	RouteCalculaTe	<ROUTE fromField="fraction_changed" fromNode="Clock"
26	27	PordoSol	<Background DEF="PordoSol" groundAngle="1.309, 1.571"
27	28	ParedeDirPendu	<Transform DEF="ParedeDir">
28	29	ParedeEsqPend	<Transform DEF="ParedeEsq">
29	30	ParedeNortePer	<Transform DEF="ParedeNorte">
30	31	PortaNortePend	<Transform DEF="PortaNorte">
31	32	KitVerticalPedul	<Group DEF="KitVertical">
32	33	CompHaste	<ProtoDeclare name="CompHaste">
33	34	Pendulum	<Transform DEF="Pendulum" center="0 2.0 0" translation="2 -0.5 0.8">
34	35	MPendulum	<OrientationInterpolator DEF="MPendulum"
35	36	PendulumAuxilia	<Transform DEF="PendulumAuxiliar" center="0.0 2.0 0.0" translation="2 -0.5 0.8"/>
36	37	BotaoExecutarE	<Transform DEF="BotaoExecutarExperimento" translation="4 3.3 0">
37	38	TextoExecutarE	<Transform DEF="TextoExecutarExperimento" translation="4 3.5 0.2">
38	39	BotaoResetPedu	<Transform DEF="BotaoReset" translation="4 2.4 0">
39	40	TextoResetPend	<Transform DEF="TextoReset" translation="4 2.6 0.2">

Figura 5.9 Exemplo dos objetos armazenados na tabela Objetos

Um ambiente virtual no VEPersonal é representado pelo identificador do ambiente (*id_ambiente*), um *nome* e uma *área* que identifica o conteúdo daquele ambiente. Estes ambientes são armazenados na Tabela *Ambiente*. A Figura 5.10 apresenta um exemplo dos ambientes do VEPersonal armazenados nessa tabela.



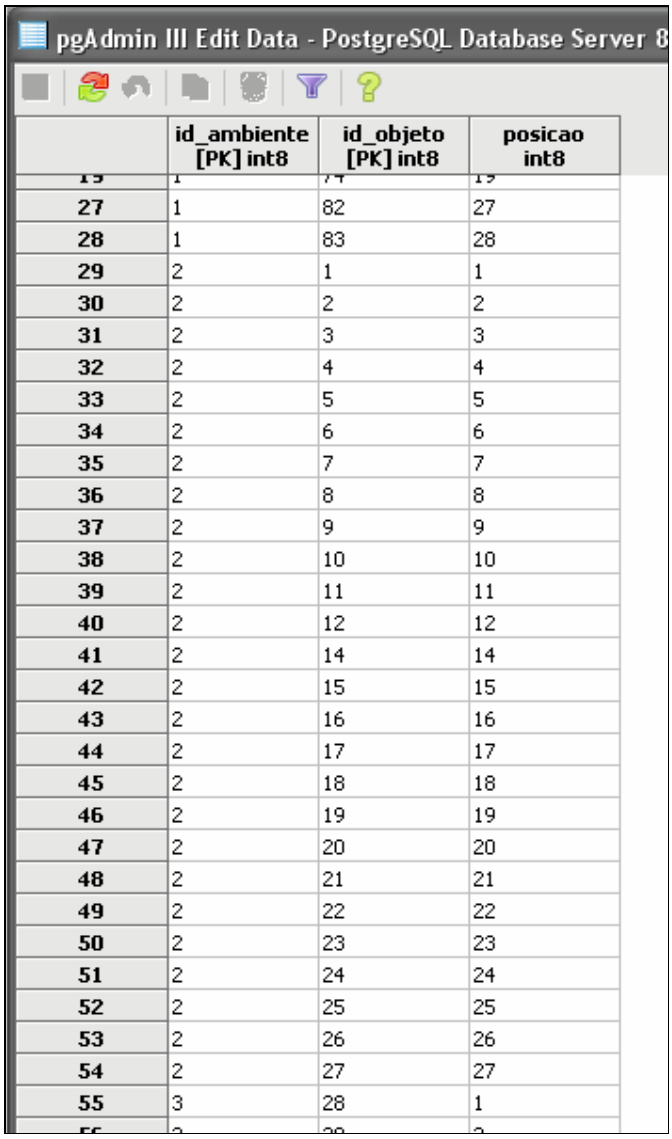
pgAdmin III Edit Data - PostgreSQL Database Serv

	id_ambiente [PK] int8	nome varchar	area varchar
1	1	hall	fisica
2	2	queda livre	fisica
3	3	pendulum	fisica
*			

Figura 5.10 Exemplo dos ambientes do VEPersonal armazenados na tabela Ambiente

A associação dos objetos a um ambiente virtual é realizada através da construção de uma tabela (tabela *Ambiente_Objetos*), relacionando o identificador do ambiente (*id_ambiente*) com a lista dos identificadores de objetos que pertencem ao ambiente (*id_objeto* - obtidos da tabela *Objetos*). Neste caso, uma associação entre a estrutura relacional e a de XML tornou possível o gerenciamento dos ambientes virtuais no VEPersonal.

Por exemplo, a estrutura de armazenamento do ambiente virtual Queda Livre pode ser visualizada na Figura 5.11. Verifica-se na coluna *id_objeto* que existe a referência aos objetos armazenados na tabela *Objetos*.

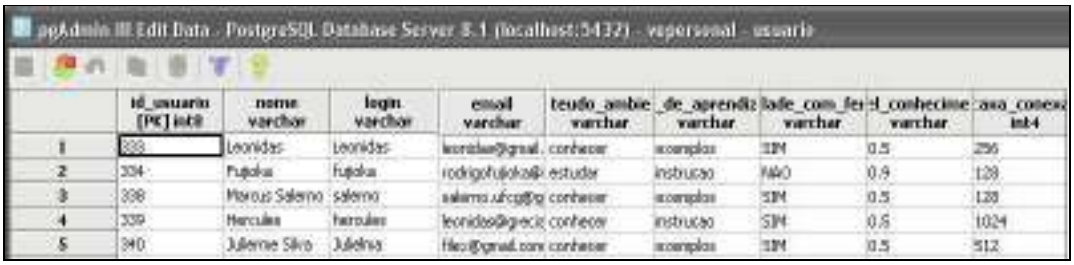


	id_ambiente [PK] int8	id_objeto [PK] int8	posicao int8
27	1	82	27
28	1	83	28
29	2	1	1
30	2	2	2
31	2	3	3
32	2	4	4
33	2	5	5
34	2	6	6
35	2	7	7
36	2	8	8
37	2	9	9
38	2	10	10
39	2	11	11
40	2	12	12
41	2	14	14
42	2	15	15
43	2	16	16
44	2	17	17
45	2	18	18
46	2	19	19
47	2	20	20
48	2	21	21
49	2	22	22
50	2	23	23
51	2	24	24
52	2	25	25
53	2	26	26
54	2	27	27
55	3	28	1

Figura 5.11 Exemplo do armazenamento do ambiente Queda Livre na tabela Ambiente_Objetos

Fazendo-se uma comparação entre as tabelas das Figuras 5.10 e 5.11, verifica-se que somente alguns objetos (os que pertencem ao ambiente Queda Livre) são referenciados. Desta forma, a construção de um AV é realizada com a recuperação dos objetos que interessam àquele ambiente. Esta solução permite o reuso do mesmo objeto em outros AV.

O cadastro do usuário possui uma estrutura relacional e é armazenado na Tabela *Usuario*. Na Figura 5.12, é apresentado um exemplo dos cadastros dos diversos usuários registrados no sistema.


pgAdmin III Edit Data - PostgreSQL Database Server 8.1 (localhost:5432) - vopersonal - usuario

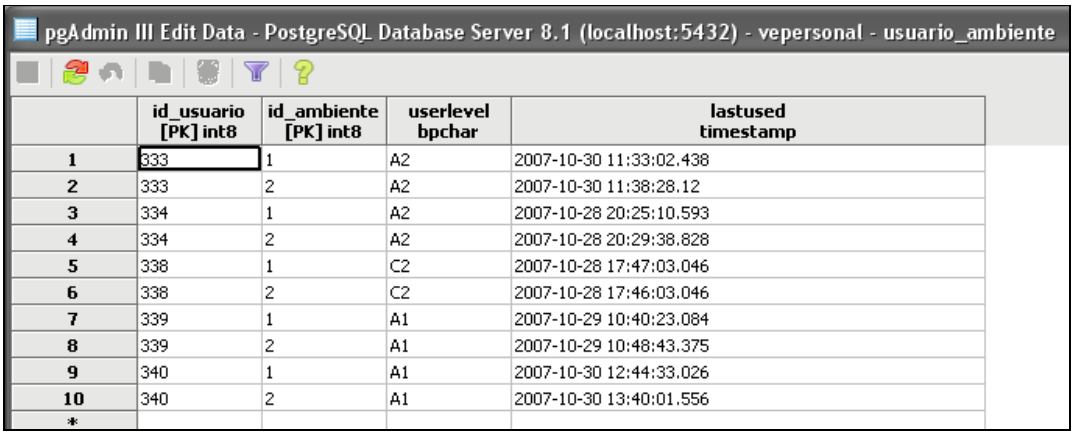
	id_usuario [PK] int8	nome varchar	login varchar	email varchar	teudo_ambie varchar	_de_aprendiz varchar	lade_com_fer varchar	l_conhecime varchar	ua_conexa int4
1	333	Leonidas	Leonidas	leonidas@gmail.	conhecer	acomplos	SIM	0.5	256
2	334	Fujioka	fujioka	rodrigofujioka@	estudar	instrucao	NAO	0.9	128
3	338	Marcus Salerno	salerno	salerno.ufcg@g	conhecer	acomplos	SIM	0.5	128
4	339	Hercules	hercules	leonidas@grecia	conhecer	instrucao	SIM	0.5	1024
5	340	Juliemie Silva	Juliehva	filez@gmail.com	conhecer	acomplos	SIM	0.5	512

Figura 5.12 Exemplo de armazenamento dos dados do usuário na tabela Usuario

O sistema também mantém uma tabela com as informações dos ambientes utilizados pelos usuários (Tabela *Usuário_Ambiente*). Esta tabela é necessária para identificar os ambientes, com seus respectivos níveis de detalhes, que foram utilizados por um determinado usuário. O objetivo é poder apresentar o ambiente atualizado em função da última visita do usuário. Com esta tabela, o sistema pode, por exemplo, recuperar um ambiente já visitado pelo usuário ou, caso haja uma interrupção de conexão, recuperar o último ambiente que o usuário estava interagindo.

Na Figura 5.13, é apresentado um exemplo dos ambientes visitados pelos usuários. Cada vez que um ambiente é carregado pelo sistema, o identificador (*id_ambiente*) é associado ao usuário (*id_usuario*) que está interagindo e ao seu respectivo *userLevel* (necessário para a identificação do nível de detalhes que o ambiente possui). Além disso, foi incluído um atributo (*lastused*) que determina quando (data e hora) o usuário acessou o referido ambiente.

Uma vez definidas a estrutura lógica do Banco de Dados e a forma de armazenamento dos ambientes virtuais e seus respectivos objetos, serão discutidas a seguir, as consultas realizadas pelos agentes para acessar os dados.



	id_usuario [PK] int8	id_ambiente [PK] int8	userlevel bpchar	lastused timestamp
1	333	1	A2	2007-10-30 11:33:02.438
2	333	2	A2	2007-10-30 11:38:28.12
3	334	1	A2	2007-10-28 20:25:10.593
4	334	2	A2	2007-10-28 20:29:38.828
5	338	1	C2	2007-10-28 17:47:03.046
6	338	2	C2	2007-10-28 17:46:03.046
7	339	1	A1	2007-10-29 10:40:23.084
8	339	2	A1	2007-10-29 10:48:43.375
9	340	1	A1	2007-10-30 12:44:33.026
10	340	2	A1	2007-10-30 13:40:01.556
*				

Figura 5.13 Exemplo da relação entre ambiente e usuário na tabela Usuario_Ambiente

Para possibilitar a integração dos comandos executados pelos agentes (escritos em Java) às consultas realizadas no SGBD, foi utilizada a API JDBC (*Java Database Connectivity*) [JDBC, 2007]. Esta API permite o acesso ao PostgreSQL, através da qual são executadas as consultas SQL e XPath.

Com o objetivo de evitar a criação de uma dependência das regras de negócio dos agentes em relação ao código de acesso dos dados, gerando um alto acoplamento entre eles, a solução foi abstrair e encapsular o acesso ao Banco de Dados através de um padrão de projeto denominado *Data Access Object* (DAO) [DAO, 2007]. Um DAO implementa o mecanismo de acesso para se trabalhar, por exemplo, com consultas a um SGBD específico. As regras de negócio são utilizadas apenas na interface do DAO, que esconde toda a complexidade existente na interação com a estrutura dos dados do SGBD. Além disso, por abstrair o acesso ao Banco de Dados, o uso do padrão DAO facilita a migração do sistema para outro SGBD, caso necessário.

A recuperação dos dados é realizada pelos agentes que têm as seguintes funções: o Agente Atualizador é responsável por recuperar os objetos X3D e montar o ambiente virtual; o Agente Pessoal por recuperar o perfil do usuário; e o Agente Ambiente responsável por recuperar a estrutura do ambiente que o usuário está interagindo (conforme descrito na seção 4.6).

O Agente Atualizador recupera os códigos dos objetos X3D através das informações fornecidas pelo Agente Pessoal (*pPerfil*) e pelo Agente Ambiente (*pSala*).

Agente Atualizador recupera a lista de objetos (pObjetos) do novo ambiente, consultando a Tabela Ambiente_Objetos. Em seguida, recupera os respectivos códigos

X3D consultando a Tabela Objetos e construindo o ambiente a ser apresentado ao usuário. Na Figura 5.14, é apresentado um exemplo dessa recuperação.

```
public class AgenteAtualizador extends Agent {
   // gerar novo ambiente
   public void definirAmbiente(PerfilUsuario pPerfil, Sala pSala) {
      pObjetos = VEPersonalDAO.getAmbienteObjetos (pPerfil, pSala);
      enviroment = new Ambiente();
      enviroment.setId_sala(pSala.getId_sala());
      StringBuffer sb = new StringBuffer();
      ArrayList list = (ArrayList) pObjetos.getLista_objetos();
      for(int i=0;i<list.size();i++) {
         ObjetoAmbiente oa = (ObjetoAmbiente) list.get(i);
         sb.append(oa.getCodigo_fonte());
      }
      enviroment.setCodigo_x3d(sb.toString());
      int userPort = 4456;
      this.sendX3dToInterface(userPort,enviroment.getCodigo_x3d());
   }
```

Figura 5.14 Consulta gerada pelo Agente Atualizador ao SGBD

O Agente Pessoal recupera o perfil do usuário fornecendo a identificação do usuário (pId_usuario) e o conteúdo que está sendo utilizado (pId_conteudo), através de uma consulta à Tabela *Usuario*. Um exemplo dessa consulta é apresentado na Figura 5.15.

```
public class AgentePessoal extends Agent {
  // recupera perfil do usuário
  public PerfilUsuario getPerfil(int pId_usuario, int pId_conteudo){
     return VEPersonalDAO.isPerfilCadastrado(pId_conteudo,
        pId_usuario);
  }
```

Figura 5.15 Consulta gerada pelo Agente Pessoal ao SGBD

O Agente Ambiente recupera a estrutura do ambiente, consultando a Tabela Ambiente_Objetos. Ele fornece o identificador do usuário (*pId_usuario*) e o ambiente que o usuário está interagindo (p*id_ambiente*), retornando a lista dos objetos do ambiente. Caso seja o primeiro acesso do usuário, a consulta retorna *null* (lista vazia),

pois não existe nenhum ambiente utilizado pelo usuário. A consulta do Agente Ambiente ao SGBD é apresentada na Figura 5.16.

```
public class AgenteAmbiente extends Agent {
   // recuperar ambiente visitado pelo usuário
   // retorna null caso não exista ambiente visitado
   public ObjetosAmbiente getAmbiente(int pId_Sala, int pId_usuario){
      return VEPersonalDAO.getAmbienteObjetos (perfilUsuario,
         pId_Sala);
   }
```

Figura 5.16 Consulta gerada pelo Agente Ambiente ao SGBD

A manutenção dos dados no SGBD é realizada pelo sistema Administrador. Este *framework* é utilizado como aplicação *stand-alone* e foi concebido apenas para permitir a inserção dos ambientes virtuais no SGBD.

5.3 Controle do Ambiente realizado pelos Agentes

Conforme discutido nos capítulos anteriores, a função dos agentes é gerar ambientes virtuais em função do perfil do usuário e atualizá-los de acordo com as ações realizadas pelo usuário.

Nesta seção, são apresentadas as soluções desenvolvidas para determinar as ações dos agentes durante a interação do usuário, dando ênfase aos conceitos utilizados pelos agentes, aos seus comportamentos e à comunicação realizada entre eles.

5.3.1 Ontologia utilizada pelos Agentes

A ontologia de uma aplicação multi-agente está definida como um conjunto de conceitos, ações e predicados. Esta definição é necessária para que um entendimento comum sobre os objetos do domínio seja compartilhado entre os agentes. Assim, tais informações são interpretadas da mesma forma por todos os agentes dentro da sociedade, permitindo o desenvolvimento da comunicação através de protocolos que determinam a seqüência de mensagens a serem trocadas entre os agentes durante a execução das tarefas do sistema.

Conceitos

Os conceitos são representações das entidades do domínio conforme elas são tratadas pelos agentes no VEPersonal. Esses conceitos são enviados pelos agentes como informação necessária para a execução de uma ação ou retornados como resultados de ações efetuadas.

Na Figura 5.17, são descritos os conceitos utilizados pelos agentes para uma aplicação do Ensino de Física, conforme apresentado na seção 5.1. O conceito **Conteúdo**, por exemplo, representa um assunto no qual se baseiam os experimentos, tais como Cinemática ou Dinâmica. O ambiente do experimento é dividido em Salas.

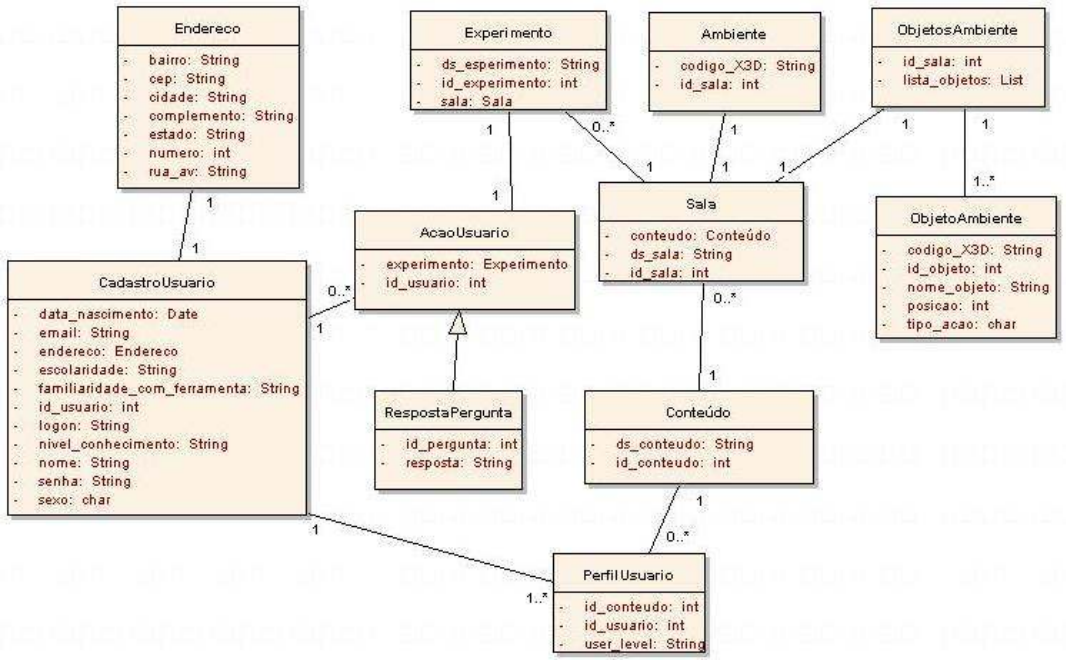


Figura 5.17 Conceitos utilizados pelos agentes para uma aplicação do Ensino de Física

O conceito **Sala**, com seus respectivos experimentos, está associado a um único **Conteudo**. Cada **Conteudo** pode estar associado a uma ou mais salas.

De acordo com o conteúdo de cada sala, são criados os **Experimentos**, que visam o ensino desse **Conteudo**. Um experimento da sala cujo conteúdo é Cinemática poderia ser, por exemplo, "Queda Livre", outro poderia ser "Trajetória de uma Partícula".

O perfil do usuário é representado pelo conceito **PerfilUsuario**. Esse conceito relaciona um usuário ao atributo *userLevel*. Este representa o perfil do usuário para um

determinado ambiente. Com base neste *userLevel*, o sistema realiza as adaptações do ambiente.

Cada **PerfilUsuario** está associado a um único **Conteudo** e cada **Conteudo** pode estar associado a vários **PerfilUsuario**, obedecendo a restrição de ser um **Conteudo** por usuário.

Cada usuário possui um cadastro que foi preenchido na primeira vez que ele acessou o sistema. Esse cadastro está representado pelo conceito **CadastroUsuario**. Sendo assim, um **PerfilUsuario** se associa a um único **CadastroUsuario**.

Realizado o cadastro, é gerado um ambiente virtual para o usuário. Neste ambiente, o usuário começa a interagir, manipulando objetos, executando experimentos e respondendo a perguntas. A interface monitora essa interação e repassa para o Agente Gerente as ações do usuário. As ações sempre estão associadas a um usuário e a um experimento. Essas ações são representadas pelo conceito **AcaoUsuario**, que possui o *id* do usuário e um Experimento.

Para este conceito, estão associados sub-conceitos que descrevem os diversos tipos de ações do usuário. Isto permite adicionar sub-conceitos posteriormente, caso seja necessária a inclusão de novas ações do usuário.

Um sub-conceito derivado do conceito **AcaoUsuario** é **RespostaPergunta**, que representa a resposta dada pelo usuário a uma pergunta durante a execução do experimento. Quando a interface informa ao Agente Gerente, de uma ação do usuário do tipo resposta a uma pergunta, ela envia uma instância do conceito **RespostaPergunta**, que é então repassada pelo Agente Gerente ao Agente Pessoal, que por sua vez verifica se a resposta àquela pergunta leva o usuário a mudar de perfil.

Uma **AcaoUsuario** está ligada a um **CadastroUsuario** e a um **Experimento**. Cada usuário executa várias ações e cada ação está relacionada a um experimento.

O conceito **Ambiente** representa o código X3D de uma sala. Uma instância desse conceito é criada quando o usuário solicita a geração de um ambiente ou quando ele muda de sala no sistema. Em seguida, o *Agente Atualizador* envia o ambiente ao *Agente Gerente*, para que ele possa providenciar o respectivo envio à interface.

O conceito **ObjetosAmbiente** representa todos os objetos de um ambiente. Cada objeto possui o código X3D associado a ele (representado pelo atributo *codigo_X3D*), identificadores (*id_objeto* e *nome_objeto*), uma posição na qual ele deve ser colocado

no ambiente (*posicao*), e um tipo de ação a ser tomada (*tipo_acao*) – que pode assumir os valores "R" para remoção e "I" para inclusão.

O conceito **ObjetosAmbiente** representa uma lista de instâncias do conceito anterior, e está associado a uma **Sala**.

Instâncias desse conceito são criadas pelo *Agente Atualizador* em dois casos: quando é gerado um ambiente para uma nova sala, o *Agente Atualizador* passa uma instância desse conceito contendo os objetos que formam o ambiente da sala para o *Agente Gerente*; ou quando o usuário muda de perfil, o Agente Atualizador calcula os objetos que devem ser inseridos ou removidos da sala onde o usuário está e passa essa informação para o *Agente Gerente*. No caso de remoção, o atributo *tag_X3D* de cada objeto da lista não é preenchido.

O *Agente Ambiente* retorna uma instância de **ObjetosAmbiente** quando o *Agente Gerente* pergunta pela referência ao estado atual do ambiente de uma sala. O *Agente Gerente,* de posse dessa informação, a coloca na solicitação de geração de ambiente para o *Agente Atualizador*.

Ações

As ações definidas na ontologia do *VEPersonal* correspondem às operações que os agentes solicitam uns aos outros. Cada ação tem como atributos as informações necessárias para a execução da tarefa por parte do agente. O agente que solicita a ação é responsável por prover essas informações, alocando-as nos devidos atributos.

O *Agente Gerente*, exercendo o papel de coordenador de ações, sempre solicita aos demais agentes a execução das tarefas necessárias para que os objetivos do sistema sejam alcançados. Neste caso, ele nunca recebe uma solicitação para realização de alguma ação por outro agente do sistema.

O *Agente Atualizador* deve sempre calcular ou recalcular o ambiente para o usuário. Para tanto, ele precisa receber as seguintes informações: o perfil do usuário para o qual o ambiente está sendo gerado (uma instância do conceito **PerfilUsuario**) e a sala que terá seu ambiente gerado (uma instância do conceito **Sala**).

Quando o usuário entra no sistema (acessando a sala *default* de entrada no ambiente) ou muda de sala, o *Agente Gerente* solicita ao *Agente Ambiente* o ambiente em questão. Sendo assim, o fluxo pode seguir por dois caminhos:

- Se o usuário já visitou a sala, o *Agente Ambiente* retorna o nome dos objetos da sala para o *Agente Gerente*. Em seguida, o *Agente Gerente* solicita ao *Agente Atualizador* que seja gerado o ambiente para o usuário, com base no perfil, na sala e nos objetos da sala. Essas informações são passadas em uma instância da ação **GerarAmbienteCadastrado**; e
- Se o usuário não tinha visitado anteriormente a sala, o *Agente Ambiente* retorna uma mensagem de falha na execução da solicitação para o *AgenteGerente*, que então solicita ao *Agente Atualizador* que seja gerado o ambiente para o usuário, com base no perfil e na sala. Essas informações são passadas em uma instância da ação **GerarAmbienteNovo**.

Em ambos os fluxos, o que é retornado para o *Agente Gerente* depoios da execução da ação é o ambiente montado, uma instância do conceito **Ambiente**.

Em seguida, após a atualização do ambiente, o *Agente Gerente* solicita ao *Agente Ambiente* que o novo ambiente seja guardado, criando uma instância da ação **GuardarAmbienteCompleto**, contendo o perfil do usuário, a sala e os nomes dos objetos do ambiente.

Na mudança de perfil, após o *Agente Atualizador* ter calculado os novos objetos, o *Agente Gerente* solicita ao *Agente Ambiente* a execução da ação **GuardarAmbienteMudancaPerfil**, passando como atributos da ação o perfil do usuário, a sala e os objetos a serem incluídos ou removidos do ambiente. Com essas informações, o *Agente Ambiente* atualiza seu modelo do ambiente para o usuário.

O *Agente Pessoal* é responsável por manter os dados do cadastro do usuário e seus perfis, devendo: incluir os dados do usuário na base de dados quando o usuário solicita um cadastro no sistema; e alterar os dados do usuário na base de dados quando o usuário muda seus dados via sistema.

Esses dois comportamentos são representados pelas ações **CadastrarNovoUsuario** e **AtualizarCadastroUsuario**, respectivamente. Ambas as ações possuem uma instância do conceito **CadastroUsuario**.

Quando o usuário executa alguma atividade que pode levá-lo a uma mudança de perfil, o *Agente Gerente* solicita ao *Agente Pessoal* que verifique se o perfil deve ser atualizado, sendo representado pela ação **AtualizarPerfilUsuario** na ontologia. A ação possui como atributos o *id* do usuário e a atividade que ele realizou.

Recebendo uma solicitação para executar essa ação, o *Agente Pessoal* utiliza suas regras de inferência para determinar se a atividade executada pelo usuário leva a uma atualização de perfil ou não. Em caso positivo, o agente atualiza o perfil do usuário e retorna para o *Agente Gerente* o novo perfil. Em caso negativo, ele retorna uma mensagem de falha.

Predicados

Os predicados definidos na ontologia do *VEPersonal* são utilizados para suprir algumas necessidades que surgiram na especificação da comunicação entre os agentes. Os predicados são usados em mensagens enviadas pelo *Agente Gerente* ao *Agente Pessoal* ou ao *Agente Ambiente*, solicitando a recuperação de algum tipo de informação. Essas mensagens possuem a performativa QUERY_REF, que indica que o agente que enviou a mensagem está pedindo que seja recuperada a referência para algum objeto.

No caso do uso dos predicados nessas mensagens, o conteúdo de cada mensagem é uma expressão referencial na linguagem *Semantic Language* (SL) [SL, 2007], especificada pela FIPA, que pergunta qual o objeto que atende ao predicado.

Por exemplo, para perguntar ao *Agente Pessoal* se um dado *login x* e uma dada *senha y* correspondem a algum cadastro de usuário no sistema, o *Agente Gerente* constrói uma expressão referencial em SL que representa a seguinte pergunta: "Qual é o **CadastroUsuario** para o qual a *senha* é *y* e o *login* é *x*?". Essa relação entre **CadastroUsuario**, *login* e *senha* é representada pelo predicado **IsLoginCadastrado**. Caso encontre o cadastro solicitado, o agente retorna-o. Caso contrário, uma mensagem de falha é retornada informando que não foi possível atender à solicitação.

O predicado **isPerfilCadastrado** é usado de forma similar pelo *Agente Gerente* quando ele necessita que o *Agente Pessoal* recupere o perfil de um determinado usuário para um determinado conteúdo.

Por fim, o *Agente Gerente* faz uso do predicado **IsAmbienteCadastrado** para pedir ao *Agente Ambiente* que recupere os objetos do ambiente para um usuário e uma determinada sala.

5.3.2 Comportamento dos Agentes

Uma vez definidos os conceitos que os agentes utilizam, é possível descrever o comportamento dos agentes e o fluxo das mensagens trocadas entre eles. A troca de mensagens é regida por um conjunto de performativas de comunicação que indicam a intenção do remetente (o agente) ao enviar a mensagem. Exemplos de performativa de comunicação definidos pela FIPA são: REQUEST - o emitente está solicitando a execução de alguma ação; INFORM - o emitente está informando o resultado de alguma ação; QUERY - o emitente solicita que seja recuperada a referência a algum objeto.

O comportamento dos agentes é realizado em função das seguintes funcionalidades: cadastramento do usuário, autenticação do usuário, solicitação da geração de um novo ambiente, solicitação de um ambiente já visitado, avaliação da mudança de perfil, cálculo da mudança de perfil do usuário e não aceitação do usuário para mudança de perfil. Cada uma dessas funcionalidades é disparada pelo Agente Gerente, tendo em vista que é ele que controla as ações dos outros agentes do VEPersonal.

A Figura 5.18 apresenta o comportamento do Agente Gerente e do Agente Pessoal durante a solicitação de autenticação do usuário no sistema. A interface envia o *login* e senha do usuário ao Agente Gerente. Em seguida, uma mensagem é enviada ao Agente Pessoal solicitando a validação do *login* e senha (`Msg 1. QUERY_REF`)[9]. O Agente Pessoal consulta o Modelo do Usuário para verificar se é um usuário cadastrado e se confere o seu *login* e senha. Caso ele exista, o seu perfil é recuperado (*userLevel*) e enviado ao Agente Gerente (`Msg 2. INFORM_REF`). Caso contrário, uma mensagem informa que o *login* e senha não conferem (`Msg 2. FAILURE`).

[9] Msg i (i= 1, 2, ..., n) – indica qual é a mensagem trocada entre os agentes no diagrama de seqüência das Figuras 5.18 a 5.24.

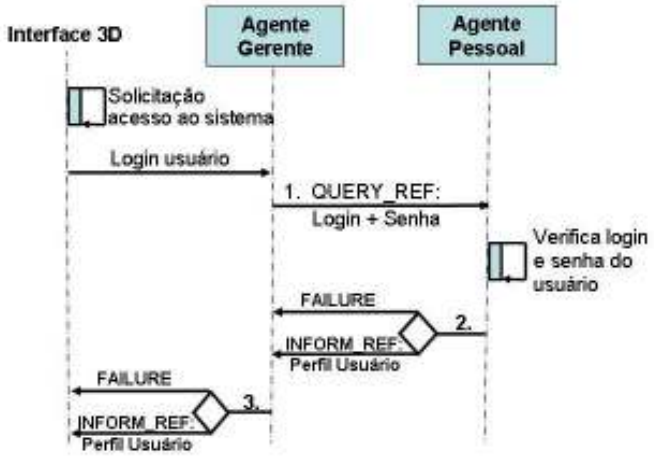


Figura 5.18 Solicitação de Acesso do Usuário ao Sistema

Se for a primeira vez que o usuário estiver acessando o sistema, ele terá que fazer um cadastro. A Figura 5.19 apresenta o comportamento dos agentes que, neste caso, é iniciado quando o usuário pressiona o botão "`CONFIRMAR`" na tela de cadastro do sistema. O cadastro é enviado pela interface para o Agente Gerente. Em seguida, uma mensagem do Agente Gerente é enviada para o Agente Pessoal solicitando a realização do cadastramento do usuário (`Msg 1. REQUEST`).

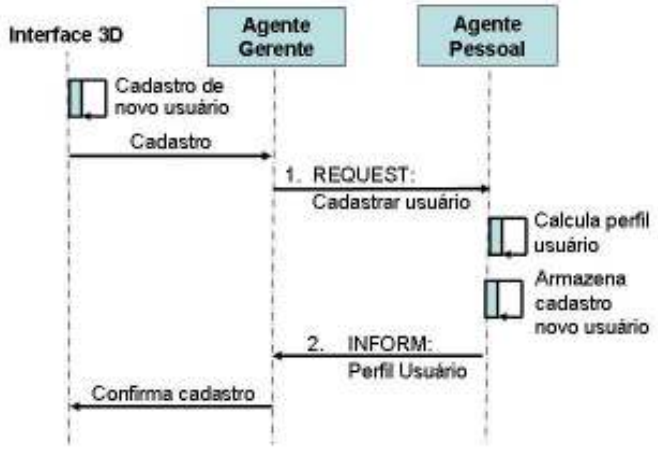


Figura 5.19 Cadastramento do Usuário

O Agente Pessoal identifica a mensagem e inicia o cálculo do perfil utilizando os dados contidos no cadastro do usuário. As regras R1 a R4 e seus respectivos Fatores de Confiança (FC), descritos na seção 4.4, são aplicados para esse cálculo.

Deve-se salientar que os valores dos FC, que são a crença que o sistema tem para as hipóteses de cada regra, variam em função da aplicação e do conjunto de regras gerados.

Por exemplo, na aplicação de Ensino de Física, se um aluno informar que está no 1º ano do ensino médio, tem menos de 15 anos e possui pouco conhecimento sobre o conteúdo a ser estudado, as regras R1, R2 e R3 (apresentadas nos Métodos de Definição de Perfil da seção 4.4) são utilizadas para determinar o FC do nível de escolaridade do aluno, da sua faixa etária e do seu nível de conhecimento sobre o conteúdo. A regra R4 (também apresentada na seção 4.4) é responsável por atualizar o novo valor do FC (FC_{new}) no cálculo do perfil do usuário. Os valores do FC para cada regra desta seção estão baseados em observações do comportamento de aprendizagem do aluno, obtidos junto a professores de ensino de Física[10].

Aplicando-se a regra R1, o FC calculado é -0,4, pois o aluno está no 1º ano do ensino médio. A regra R4 atualiza o Fator de Confiança que será utilizado para determinar o perfil do usuário. Como $FC < 0$ e FC_{old} inicialmente é 0.0, a equação dessa regra a ser aplicada é a (3). Sendo assim, o FC_{new} é:

R1: Definição do nível de escolaridade do usuário

```
Se aluno está no 1° ano do ensino médio então
  Tem nível de escolaridade para executar o experimento com FC = -0,4
Se aluno está no 2° ano do ensino médio então
  Tem nível de escolaridade para executar o experimento com FC = 0,0
Se aluno está no 3° ano do ensino médio então
  Tem nível de escolaridade para executar o experimento com FC = 0,6
```

R4: Determinação do novo Fator de Confiança

FC_{new} = (FC_{old} + FC) / (1 – min(|FC_{old}|, |FC|)), caso contrário

FC_{new} = (0,0 - 0,4) / (1 – min (|0|, |-0,4|) = -0,4

[10] Os valores do FC, utillizados nesta seção, foram aplicados para identificação do nível de conhecimento dos alunos durante a fase de testes. Entretanto, um estudo de avaliação de desempenho dos alunos pode ser realizado para propor ajustes aos valores de FC.

Na regra R2, como o usuário tem menos de 15 anos, o FC é -0,2. Aplicando-se a R4 novamente e considerando que FC < 0 e FC_{old} < 0 (pois FC_{old} = -0,4) isso implica no uso da equação (2):

R2: Definição da faixa etária do usuário

```
Se faixa etária < 15 anos
      Tem idade adequada para realizar aplicação FC = -0,2
Se 15 ≤ faixa etária < 17 anos
      Tem idade adequada para realizar aplicação FC = 0,2
Se faixa etária ≥ 17 anos
      Tem idade adequada para realizar aplicação FC = 0,6
```

R4: Determinação do novo Fator de Confiança

$FC_{new} = FC_{old} + FC\ (1 + FC_{old})$, para ($FC_{old}$ e FC <0) (2)

$FC_{new} = -0,4 -0,2 * (1 -0,4)) = -0,4 - 0,12 = -0,52$

Com a aplicação da regra R3, tem-se que FC= -0,6, pois o aluno informou que tem conhecimento sobre o conteúdo. Utilizando a regra R4, equação (2), o novo FC será FC_{new}= -0,808.

R3: Definição do nível de conhecimento do usuário sobre o conteúdo C_i (i= 1, 2, ..., n)[11]

```
Se o conhecimento sobre o conteúdo C1 = "pouco" então
      Tem conhecimento adequado para realizar a aplicação FC = -0,6
Se o conhecimento sobre o conteúdo C1 = "razoável" então
      Tem conhecimento adequado para realizar a aplicação FC = 0,2
Se o conhecimento sobre o conteúdo C1 = "muito" então
      Tem conhecimento adequado para realizar a aplicação FC = 0,6
```

R4: Determinação do novo Fator de Confiança

$FC_{new} = FC_{old} + FC\ (1 + FC_{old})$, para ($FC_{old}$ e FC <0) (2)

$FC_{new} = -0,52 -0,6 * (1 -0,52)) = -0,52 - 0,288 = -0,808$

Terminado o cálculo do novo FC em função dos dados fornecidos no cadastro do usuário, o perfil do usuário é determinado utilizando a regra R5 da seção 4.4 (lembrando

[11] Nesta aplicação podem existir vários conteúdos denominados de $C_1, C_2, ..., C_n$.

que FC_{user} recebe o valor de FC_{new}). Para este exemplo, as faixas de valores do perfil do usuário foram definidas sob o ponto de vista da capacidade de aprendizagem sobre o conhecimento estudado. O usuário é considerado iniciante se o seu FCuser < -0,4 e avançado se o FCuser > 0,4. Para os valores entre -0,4 e 0,4 o usuário é considerado com maturidade intermediária de aprendizagem.

R5: Definição do *userLevel* em função do nível de escolaridade, faixa etária e nível de conhecimento.

Se $FC_{user} \leq -0,4$ então
    userLevel = "iniciante"
Se $-0,4 < FC_{user} \leq 0,4$ então
    userLevel = "intermediario"
Se $FC_{user} > 0,4$ então
    userLevel = "avançado"

O perfil inicial encontrado para o usuário, com a aplicação da regra R5, é de um aluno "iniciante", pois seu FC_{user}= -0,808. Identificado o perfil e determinado o seu *userLevel*, esses dados são armazenados no Modelo do Usuário, que está representado na tabela *Usuário* (ver Figura 5.12).

Em seguida, o Agente Pessoal informa ao Agente Gerente que o cadastramento foi realizado com sucesso, retornando na mensagem o perfil do usuário (`Msg 2. INFORM_REF`).

Quando o Agente Gerente recebe a mensagem de confirmação, ele chama o método da interface informando que o cadastramento ocorreu com sucesso. Após isso, o usuário é habilitado a acessar os Experimentos de Física.

As Figuras 5.20 e 5.21 apresentam o fluxo de informações trocadas entre os agentes para a geração de um novo ambiente. Esse comportamento é acionado quando o usuário escolhe a opção "`CARREGAR AMBIENTE`" na tela principal do sistema.

Existem duas opções de fluxo de mensagens neste caso. Quando o usuário solicita "carregar um ambiente", este ambiente pode ser novo, ou seja, o usuário ainda não acessou o ambiente. Uma segunda possibilidade é a solicitação de um ambiente em que o usuário já tenha visitado anteriormente e, conseqüentemente, existem informações

importantes sobre o ambiente que devem ser recuperadas (por exemplo, número de interações realizadas com o experimento).

A Figura 5.20 descreve o primeiro caso, em que o usuário solicita um ambiente que ele ainda não visitou. A interface envia para o Agente Gerente essa solicitação. O Agente Gerente, inicialmente, solicita ao Agente Pessoal o perfil do usuário através de uma mensagem (`Msg 1.QUERY-REF`). Este agente consulta o SGBD e recupera o *userLevel* do usuário que é "iniciante".

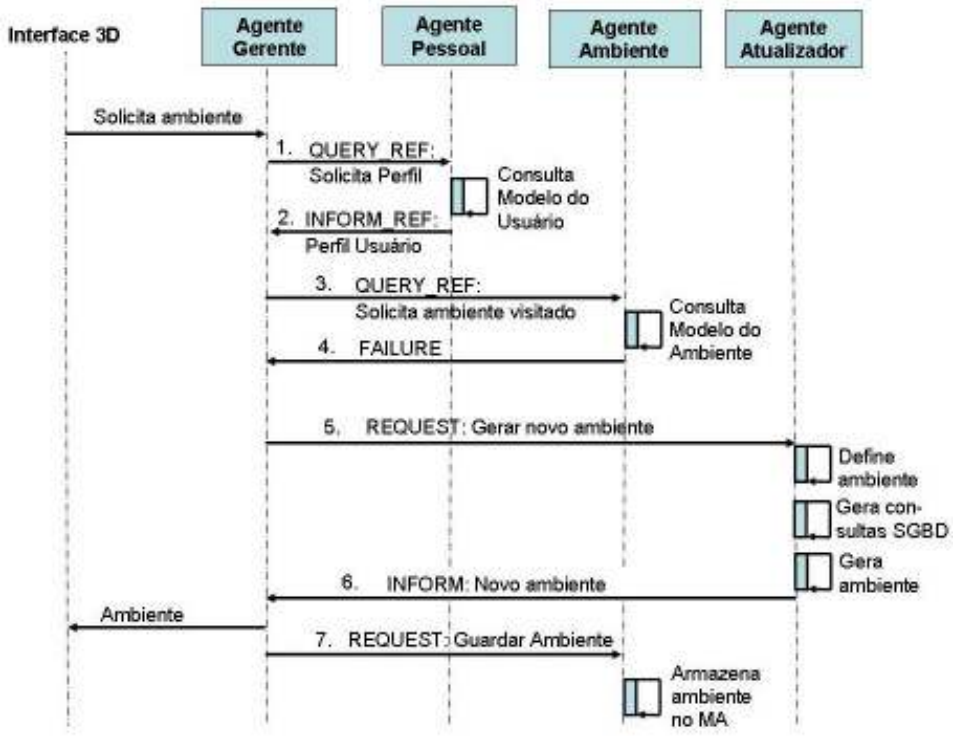


Figura 5.20 Solicitação de um Novo Ambiente

Ao receber o perfil do usuário do Agente Pessoal (`Msg 2.INFORM_REF`), o Agente Gerente pergunta ao Agente Ambiente se o usuário já visitou o ambiente solicitado. O Agente Ambiente consulta o Modelo do Ambiente para saber se tem uma associação de algum ambiente com o usuário (`Msg.3.QUERY_REF`). Como é a solicitação de um novo ambiente, a resposta é vazia, indicando que o usuário ainda não visitou aquele ambiente (`Msg 4.FAILURE`). Como visto no início desta seção, um ambiente representa uma sala que possui os experimentos de Física.

De posse dessa resposta, o Agente Gerente envia uma mensagem para o Agente Atualizador solicitando um novo ambiente (`Msg 5.REQUEST`), por exemplo, o "hall de entrada" dos Experimentos de Física. Este agente realiza os seguintes passos:

- Define ambiente – consulta a tabela *Ambiente* do SGBD (Figura 5.10) para saber a identificação do ambiente a ser recuperado e quais os objetos que pertencem a esse ambiente;
- Gera consultas ao SGBD – acessa a tabela Objetos (Figura 5.9) e recupera o código X3D dos objetos pertencentes ao ambiente. Uma consulta XPath identifica o *userLevel* de cada objeto e recupera apenas a parte do código que interessa à consulta; e
- Gera ambiente – constrói o ambiente com os códigos X3D dos objetos, inserindo o cabeçalho do arquivo X3D.

Em seguida, uma mensagem do Agente Atualizador é enviada para o Agente Gerente informando o sucesso da operação e anexando o código X3D do ambiente gerado (`Msg 6.INFORM_REF`). O Agente Gerente envia o ambiente para a interface que, por sua vez, envia para o *browser* do usuário. O Agente Gerente também envia uma mensagem ao Agente Ambiente com os objetos que compõem o ambiente para serem armazenados no Modelo do Ambiente (`Msg 7.REQUEST`). O armazenamento no Modelo do Ambiente é realizado pelo Agente Ambiente através da inclusão dos dados na tabela Usuário_Ambiente (Figura 4.8) que contém a identificação do usuário (*id_usuario*), a identificação do ambiente (*id_ambiente*) e o *userLevel* do usuário.

A diferença na geração de um ambiente já visitado pelo usuário está na resposta do Agente Ambiente quando consultado pelo Agente Gerente (`Msg 4. INFORM_REF`). A resposta desse agente é a lista de objetos do ambiente, juntamente com a identificação do usuário e a identificação do ambiente (Figura 5.21).

Em seguida, o Agente Gerente envia essas informações ao Agente Atualizador (`Msg 5.REQUEST`) para que ele gere a consulta ao SGBD, recupere os objetos e monte o arquivo X3D. Este procedimento é o mesmo que o realizado na geração de um novo ambiente, contudo não é necessário a definição do ambiente, pois ele já existe.

Uma vez gerado o arquivo X3D do ambiente, esta informação é devolvida ao Agente Gerente (`Msg 6.INFORM_REF`) que a envia para a interface.

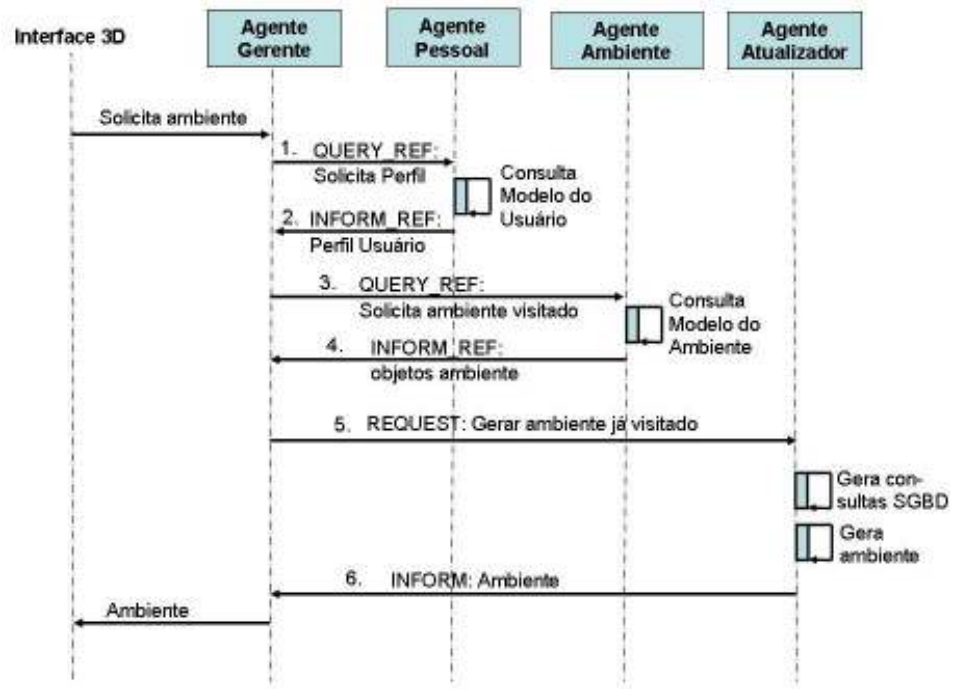


Figura 5.21 Solicitação de um Ambiente já visitado

A avaliação de Mudança de Perfil é sempre realizada pelos agentes quando o usuário executa ações que podem implicar na evolução do seu conhecimento. Na Figura 5.22, a interface envia para o Agente Gerente as ações do usuário. Para cada ação, este agente determina o tipo da ação (por exemplo, resposta correta de um teste durante a interação com o experimento Queda Livre) e envia uma mensagem para o Agente Pessoal solicitando uma avaliação para atualização de perfil (`Msg 1.REQUEST`). Ao receber a mensagem, o Agente Pessoal utiliza a regra R6 para calcular o FC da ação do usuário e aplicar as regras R4 e R5 para avaliar o seu perfil (ver seção 4.4).

R6 – Determinação do FC_{know} e $FC_{notKnow}$ em função do acerto ou erro do teste T_i (i= 1, 2, ..., n)[12]

```
Se a resposta do teste T1 está correta então
      Usuário sabe o conteúdo C1 com FCknow =  0,6
Caso contrário
      Usuário não sabe o conteúdo C1 com FCnotknow =  0,4
```

[12] Para cada conteúdo do ambiente, podem existir diversos testes, denominados neste trabalho de T_1, T_2, ...T_n.

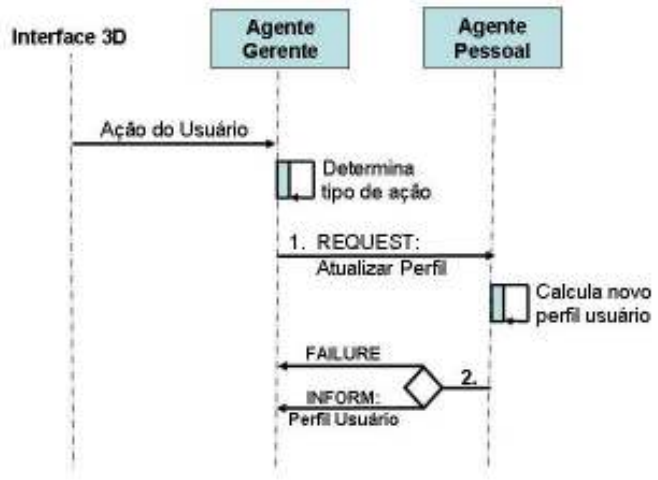


Figura 5.22 Avaliação de Mudança de Perfil

Como o usuário acertou o teste, de acordo com a regra R6 o seu FC será 0,6. Sendo o FC inicial do usuário FC_{old} = -0,808 (FC_{old} < 0) e o FC = 0,6 (FC > 0), aplica-se a equação (3) da regra R4 para calcular a atualização do perfil:

R4 – Determinação do novo Fator de Confiança

$$FC_{new} = (FC_{old} + FC) / (1 - min(|FC_{old}|, |FC|)), \text{ caso contrário} \quad (3)$$

$$FC_{new} = (-0,808 + 0,6) / (1 - min(|-0,808|, |0,6|)) = -0,52$$

Verifica-se que houve um aumento do FC do usuário, o que significa que o usuário teve um ganho na sua capacidade cognitiva, mas ainda não é suficiente para que o sistema decida pela mudança de seu perfil. Neste caso, o conteúdo da resposta do Agente Pessoal é nulo e a performativa enviada é FAILURE (`Msg 2.FAILURE`).

Entretanto, caso o usuário responda a novos testes corretamente, o FC_{new} irá aumentar e poderá ultrapassar o valor de FC_{new} > -0,4, justificando uma mudança de nível para *userLevel* = "intermediário". Neste momento, o Agente Pessoal retorna ao Agente Gerente uma mensagem, cujo conteúdo é o novo perfil do usuário (`Msg 2.INFORM_REF`).

Uma vez recebida a mensagem de alteração do perfil do usuário, o Agente Gerente envia, através da interface, uma mensagem ao usuário informando que houve mudança do seu perfil e que o ambiente vai ser atualizado. É dada a opção ao usuário de

aceitar a mudança ou não. Este procedimento é realizado para que o usuário tenha conhecimento que irá ocorrer mudanças no ambiente.

Caso ele aceite (ver Figura 5.23), o Agente Gerente solicita ao Agente Ambiente o ambiente atual que o usuário está interagindo (`Msg 1.QUERY_REF`). O Agente Ambiente consulta o Modelo do Ambiente, através da tabela *Ambiente* (ver Figura 5.10), recupera a lista de objetos do ambiente, utilizado pelo usuário, e a envia para o Agente Gerente (`Msg 2.INFORM_REF`). Este agente elabora uma mensagem ao Agente Atualizador solicitando a geração do ambiente com o novo perfil (`Msg 3.REQUEST`).

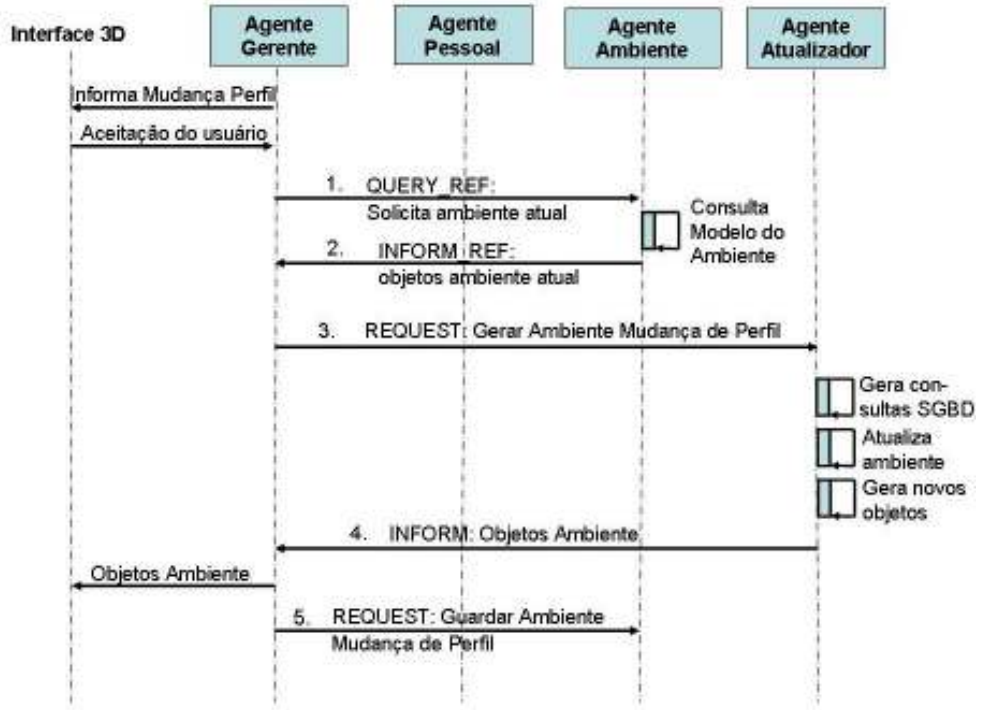


Figura 5.23 Geração de um novo Ambiente em função da Mudança de Perfil

A função do Agente Atualizador, neste caso, é pesquisar no SGBD quais os objetos da lista que possuem o novo perfil definido para o usuário (no caso, o novo perfil é "intermediário"). Os objetos encontrados para este perfil são recuperados. Uma mensagem retornando a lista dos nomes dos objetos e os respectivos códigos X3D é enviada ao Agente Gerente (`Msg 4.INFORM_REF`). Em seguida, o Agente Gerente envia essa lista à interface.

O objetivo dessa lista de objetos (nome + código X3D) é permitir que a interface realize a substituição apenas dos objetos que necessitem ser modificados no ambiente.

Desta forma, não é necessário carregar todo o ambiente em função da mudança do perfil, diminuindo a sobrecarga no processo de transmissão de dados e evitando a reinicialização da cena. Na seção 5.4, o processo de atualização do ambiente na interface de comunicação cliente/servidor é demonstrado através de exemplos de interação com o referido usuário.

A atualização do ambiente também é informada ao Agente Ambiente (`Msg 5.REQUEST`). Uma lista com o nome dos objetos é enviada para este agente que irá atualizar o Modelo do Ambiente.

A Figura 5.24 descreve o caso em que o usuário não aceita mudança no seu perfil. Esta possibilidade foi implementada porque pode haver casos em que não interessa ao usuário alterar o seu ambiente (talvez porque ele queira completar a sessão com o perfil atual e conhecer melhor todo o processo daquele nível, por exemplo).

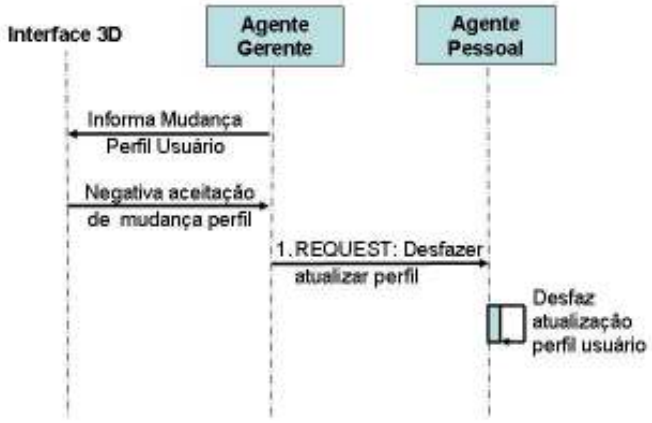


Figura 5.24 Não aceitação de mudança do Perfil

Então, se o usuário não aceitar a alteração do perfil (“Negativa de aceitação de mudança”), o Agente Gerente comunica ao Agente Pessoal que ele deve desfazer a atualização do perfil (`Msg 1.REQUEST`) e retornar ao perfil anterior.

Neste caso, o comportamento do Agente Gerente termina assim que ele envia a mensagem para o Agente Pessoal.

5.3.3 Comunicação entre os Agentes

A comunicação entre os agentes foi desenvolvida utilizando-se a plataforma JADE, a linguagem Java e o ambiente Eclipse [Eclipse, 2007]. A validação dessa comunicação foi realizada através de ferramentas de testes do JADE, que permitem acompanhar os comportamentos dos agentes e as mensagens trocadas entre si.

Uma dessas ferramentas, o *Sniffer Agent* [Bellifemine et al., 2001], monitora a troca de mensagens entre os agentes, apresentando a seqüência dessas mensagens em tempo de execução, além de ser possível visualizar o conteúdo de cada uma.

Como exemplo, na Figura 5.25, pode-se verificar a seqüência de mensagens trocadas entre os agentes quando o usuário muda de sala (novo ambiente ou para um outro que ele já havia visitado anteriormente). No lado esquerdo da figura, estão os agentes pertencentes ao VEPersonal. No lado direito, aparecem as mensagens trocadas entre os agentes com a respectiva ordem de execução. Em cada mensagem, aparece a performativa de comunicação e o identificador (id) da conversação.

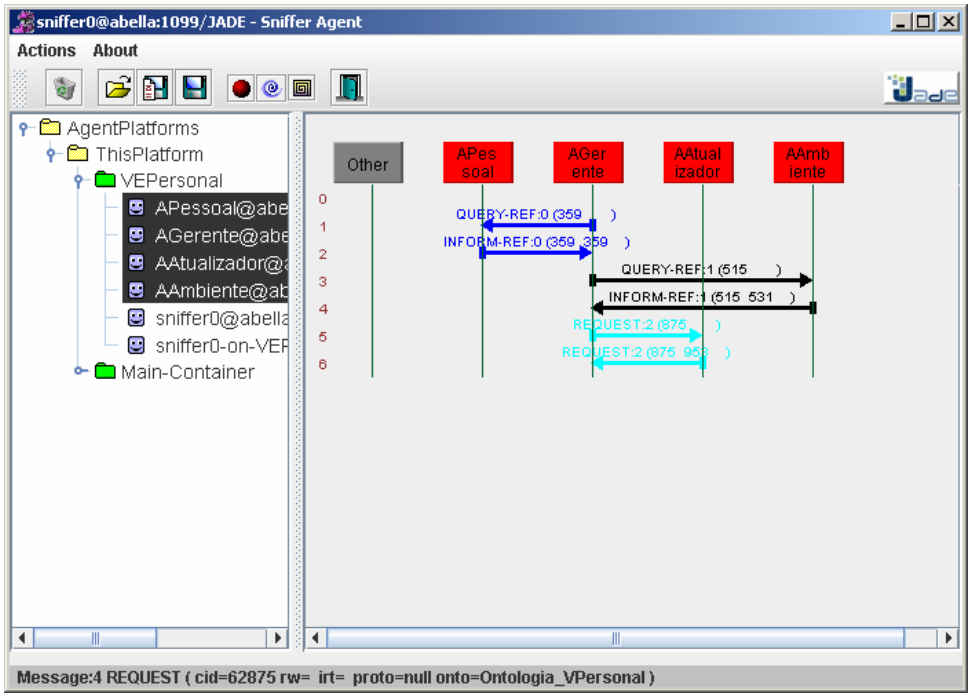


Figura 5.25 Visualização da troca de mensagens entre os agentes

No ambiente JADE, as mensagens utilizam as performativas de comunicação especificadas pela FIPA-ACL [FIPA-ACL, 2007], das quais pode-se verificar: o agente

que enviou a mensagem (*Sender*), o agente que recebeu (*Receivers*), o conteúdo da mensagem (*Content*), a ontologia (*Ontology*) e o tipo de ação a ser realizada (*Conversation-id*).

A estrutura de uma mensagem trocada entre os agentes pode ser visualizada na Figura 5.26. Neste exemplo, apresenta-se a mensagem enviada pelo Agente Gerente para o Agente Atualizador solicitando os objetos X3D para ser construído o ambiente do usuário.

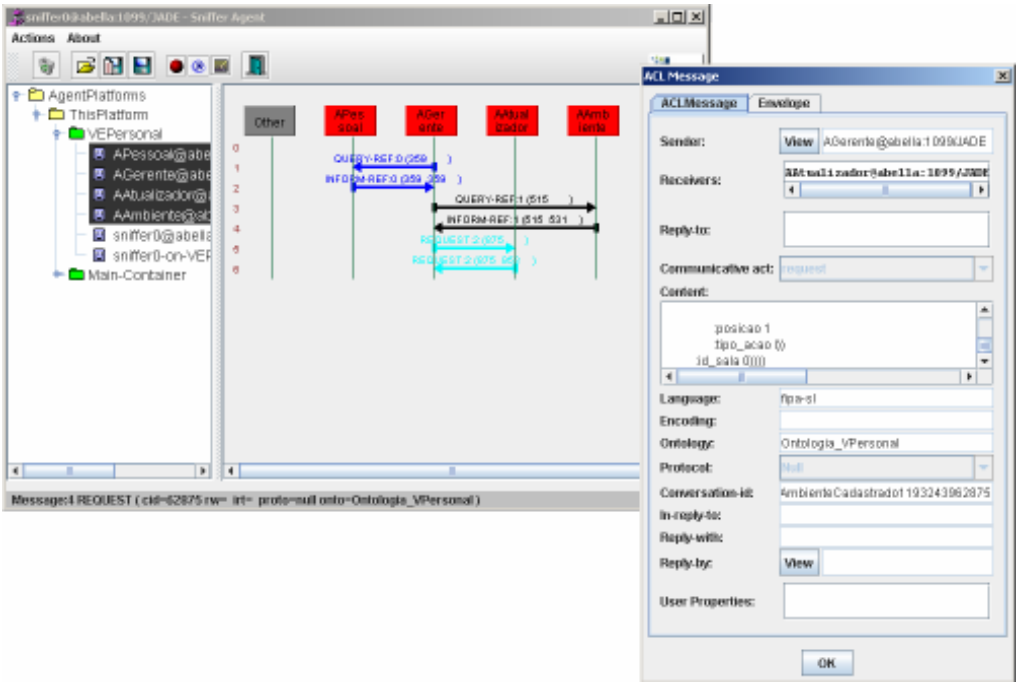


Figura 5.26 Estrutura da mensagem enviada pelo Agente Gerente ao Agente Atualizador

Na Figura 5.27, é apresentado o conteúdo da mensagem que o Agente Atualizador retorna em resposta à solicitação do Agente Gerente, conforme descrito na Figura 5.26. A mensagem contém a identificação do ambiente e o código X3D dos objetos correspondentes ao ambiente.

```
((result
   (action
    (agent-identifier
     :name AAtualizador@abella:1099/JADE
     :addresses
      (sequence http://192.168.254.4:7778/acc http://192.168.254.4:1056/acc))
    (GerarAmbienteUsuarioCadastrado
     :perfil_usuario
      (PerfilUsuario
       :user_level "1B"
       :id_usuario 334
       :id_conteudo 1)
     :sala
      (Sala
       :conteudo
        (Conteudo
         :id_conteudo 1
         :ds_conteudo default)
       :id_sala 1
       :ds_sala default)
     :objetos_ambiente
      (ObjetosAmbiente
       :lista_objetos
        (sequence
         (ObjetoAmbiente
          :id_objeto 1
          :nome_objeto object1
          :codigo_fonte "<?xml version=\"1.0\" encoding=\"UTF-8\"?>
<X3D profile=\"Immersive\" version=\"3.0\">
<Scene>
....
</Scene> </X3D>
```

Figura 5.27 Conteúdo da mensagem retornada pelo Agente Atualizador para o Agente Gerente

5.4 Estrutura da Interface de Comunicação Cliente/Servidor do VEPersonal

A interface de comunicação do VEPersonal foi estruturada para que a conexão cliente/servidor via Web pudesse ser executada de forma desacoplada. A Figura 5.28 apresenta a estrutura da interface e os padrões de projeto utilizados na comunicação entre o cliente e o servidor.

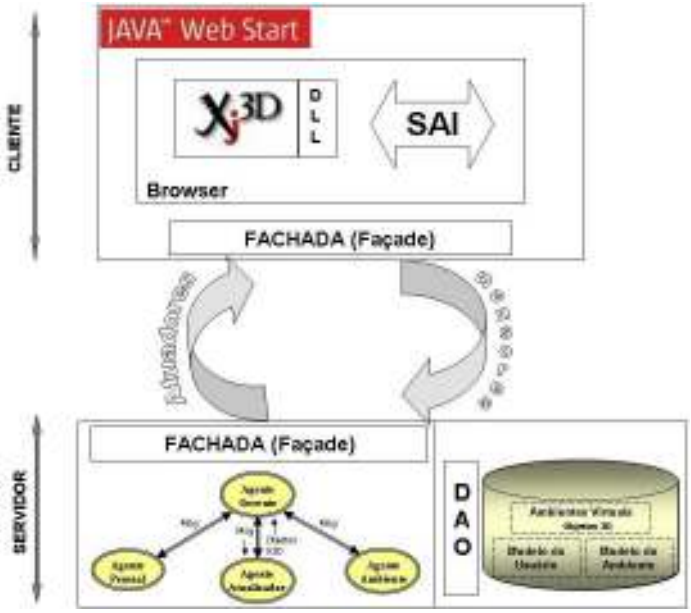


Figura 5.28 Estrutura da interface do VEPersonal e os padrões de projeto utilizados

Na máquina do cliente, o browser é gerado pelo *toolkit* Xj3D que realiza a renderização dos objetos, permitindo a visualização do ambiente virtual e a respectiva navegação e interação do usuário. Com a utilização do Xj3D é possível atualizar os ambientes virtuais através da inserção e remoção de objetos, utilizando a API *Scene Access Interface* – SAI (ver seção 4.7).

A execução do browser Xj3D necessita de diversas bibliotecas DLL (*dynamic-link library*) [DLL, 2007]. Para tornar a aplicação desacoplada, ou seja, independente da localização das bibliotecas, foi construída uma função *Load* que localiza e carrega automaticamente para a máquina do cliente as bibliotecas necessárias à aplicação. Neste caso, a aplicação pode ser executada em qualquer máquina porque não depende de bibliotecas externas.

No VEPersonal, o browser Xj3D é executado na máquina do cliente. Para que o browser possa ser acessado em ambiente Web, é utilizado o Java Web Start (JWS) que controla os serviços e protocolos de comunicação da aplicação Web (ver seção 4.7).

A Interface no VEPersonal é responsável pela comunicação da aplicação (que está sendo executada no browser) com o servidor. Esta comunicação se dá através de sensores (*sensors*) e atuadores (*effectors*). Os sensores notificam as alterações realizadas no browser, identificando o objeto e o tipo da ação executada pelo usuário. Para a implementação dos sensores foi definido um método (*Listener*), que utiliza o padrão de projeto *Observer*. Este padrão define uma dependência um-para-muitos entre objetos, de modo que, quando um objeto muda de estado, todos os seus dependentes são automaticamente notificados e atualizados. Neste caso, o *Listener* é utilizado para propagar para o servidor as mudanças e atualizações ocorridas no browser. A Figura 5.29 apresenta a implementação do método *Listener* na Interface.

Os atuadores, por sua vez, são responsáveis pelo envio dos objetos e ambientes virtuais do servidor para o browser. Tanto os atuadores como os sensores utilizam o *framework* LipeRMI [LipeRMI, 2007] para a transferência dos dados entre o cliente e o servidor, via Web.

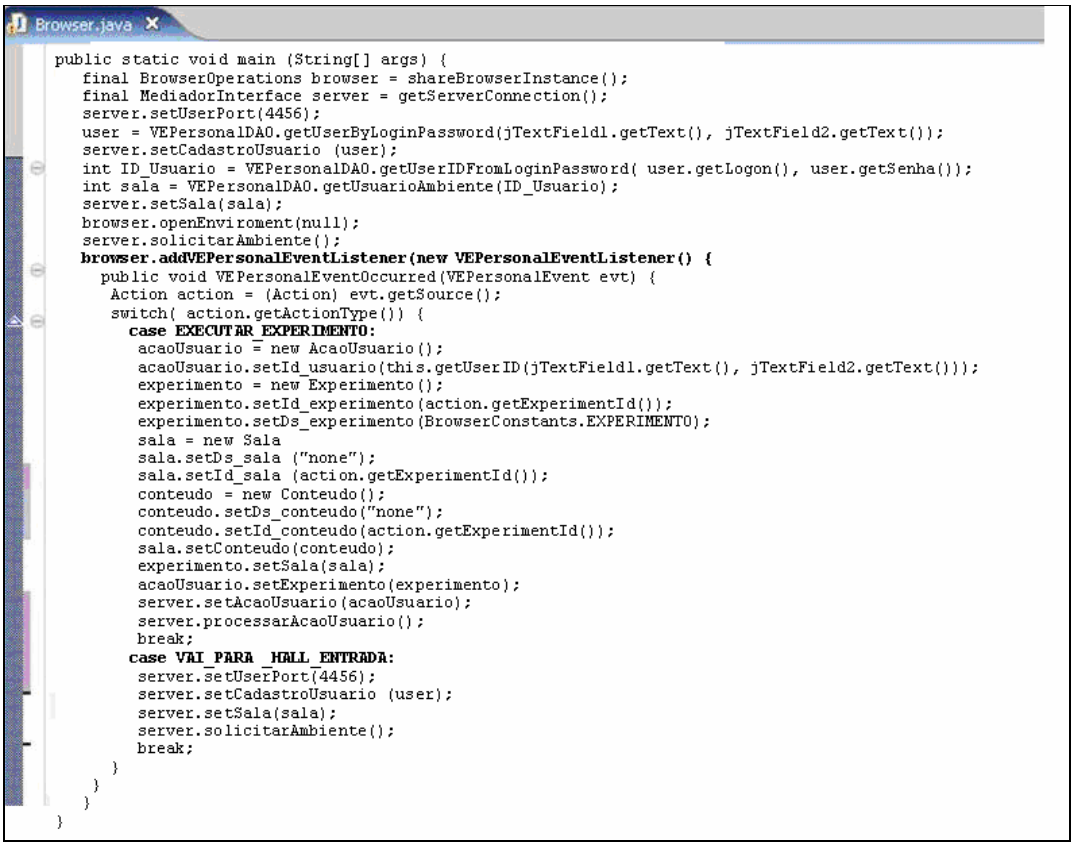


Figura 5.29 Implementação do *Listener* na Interface do VEPersonal

A proposta do LipeRMI é re-implementar o RMI [RMI, 2007] de maneira que as chamadas sejam feitas através da Internet, minimizando o uso de banda de comunicação (diminuindo o número de conexões ativas). Depois que o cliente já estiver conectado, esta conexão deve permanecer aberta durante toda sessão de troca de dados entre o cliente e o servidor (ou até que a aplicação permita que a conexão seja fechada). Desta forma, o servidor não precisa abrir uma nova conexão no sentido servidor-cliente, pois é possível utilizar a conexão já existente. E não precisando abrir novas conexões, o cliente não se preocupa com controle de *firewall* - resolvendo o maior problema do RMI.

Com o uso do LipeRMI, a comunicação é definida por eventos que ativam as conexões do cliente e do servidor, verificando quando uma conexão é iniciada e encerrada.

As Figuras 5.30 e 5.31 apresentam os métodos definidos para o comportamento do cliente e do servidor, respectivamente, durante a comunicação com a interface.

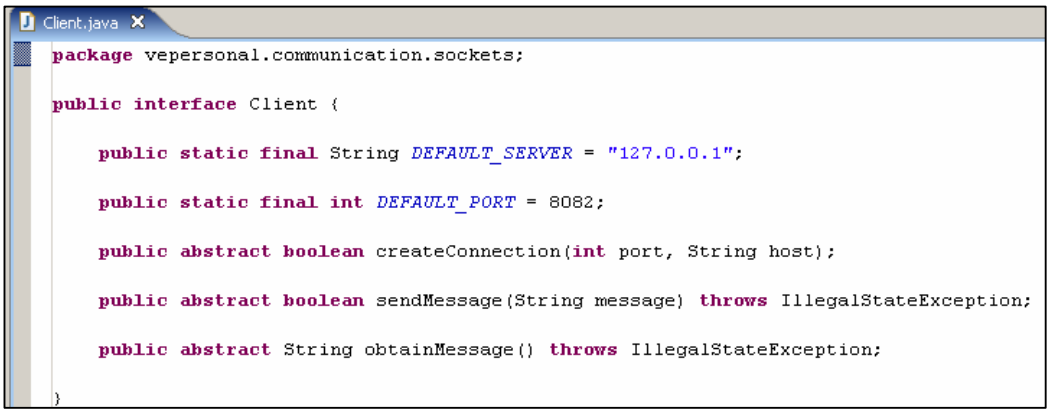


Figura 5.30 Métodos definidos para o comportamento do cliente durante a comunicação

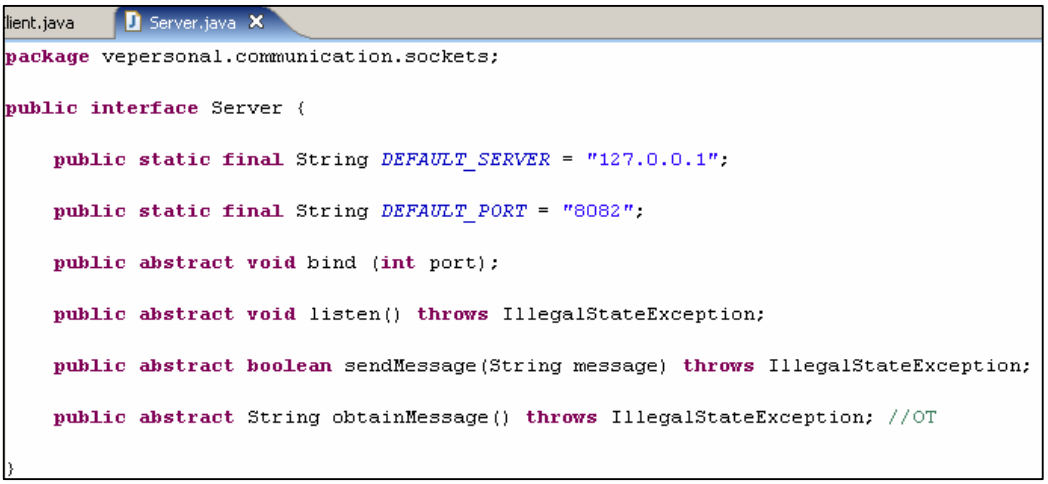


Figura 5.31 Métodos definidos para o comportamento do servidor durante a comunicação

Com o objetivo de abstrair funcionalidades do browser e diminuir o acoplamento com o servidor, tornando a comunicação independente da tecnologia empregada, criou-se uma Fachada (padrão de projeto *Facade*) que possibilita definir os padrões de comunicação do browser com a interface. *Facade* é um padrão estrutural que visa ocultar os detalhes de um processo criando uma classe de fachada, ou seja, a classe cliente apenas chama um método da classe Fachada e essa classe é que se encarrega de executar esse processo. Uma Fachada também foi desenvolvida para o servidor, especificando os padrões de comunicação com a interface. A Figura 5.32 apresenta a especificação dos métodos utilizados pela Interface para o padrão *Facade*.

```
package vepersonal.browser;
import java.util.ArrayList;
public interface BrowserFacade {
        public abstract void openEnviroment(String x3d);
        public void setEnviroment(ArrayList listaObjetos);
        public abstract String[] getObjectsList();
        public abstract boolean addObject(String objectName, String xmlObj);
        public abstract void removeObject(String x3dName);
        public abstract String getX3dEnviroment();
        public void setPerfilUsuario(PerfilUsuario perfilUsuario);
        public void setStubManager(IStubManager stubManager);
        public void removeVEPersonalEventListener(VEPersonalEventListener listener);
        public void addVEPersonalEventListener(VEPersonalEventListener listener);
        public void sairAmbiente();
}
```

Figura 5.32 Especificação dos métodos utilizados pela Interface

A Interface do VEPersonal realiza a comunicação cliente/servidor com alta coesão e baixo acoplamento devido aos padrões de projeto utilizados. Com esses padrões é possível reduzir a dependência dos serviços de conexão, gerando baixo acoplamento, e manter os serviços entre cliente e servidor focados e gerenciáveis, resultando conseqüentemente, em uma alta coesão. O uso dos padrões descritos nesta seção permite a alteração ou substituição das tecnologias empregadas, sempre que ocorrer o surgimento de novas tecnologias ou evolução das já existentes.

5.5 Desempenho do VEPersonal

O desempenho do VEPersonal foi medido levando-se em consideração o número de interações que o usuário realiza em cada sessão e o tempo de resposta que o sistema leva para executar uma atividade e/ou para realizar a adaptação necessária.

As interações são do tipo: executar um experimento, consultar informações, responder a perguntas, entrar em um novo ambiente e voltar a um ambiente já visitado. Tais interações podem ser realizadas a qualquer momento e em qualquer ordem, dependendo apenas da motivação do usuário e do seu estilo de navegação. A resposta do sistema é realizada após avaliação da ação do usuário pelos agentes e da recuperação no SGBD dos objetos que irão compor o ambiente.

Na aplicação do estudo de caso, foram desenvolvidos dois experimentos de Física: Queda Livre e Pêndulo Simples. Os testes foram realizados em uma rede local, utilizando para o servidor, um computador com processador Athlon XP de 2.4 Ghz e 512 MB de memória e para o cliente, um Notebook HP Paviloon ZE2000 com processador Intel Celeron M de 1.4 Ghz e 1GB de memória.

A versão atual do VEPersonal permite o acesso de um usuário por vez ao sistema. Para a realização dos testes, foram cadastrados inicialmente 10 usuários com seus respectivos perfis. Todos os cadastros foram armazenados no Modelo do Usuário.

Quando um usuário acessa o VEPersonal, seu *login* e senha são verificados e após essa etapa, o ambiente inicial "Hall de Entrada" é carregado (ver Figura 5.3). A partir desta etapa, os seguintes valores foram levantados em cada sessão para aferir o desempenho do sistema:

- Tempo de resposta do sistema para carregar um novo ambiente;
- Tempo de resposta do sistema para carregar um ambiente já visitado;
- Tempo de resposta do sistema para atualizar um ambiente devido à mudança de perfil.

Testes automáticos de desempenho foram gerados para avaliar o tempo gasto pelo VEPersonal para carregar os ambientes. Foi utilizado o *framework* JUnitPerf [JUnitPerf, 2006] que permite construir objetos que recebem testes existentes do JUnit e acrescentam neles a avaliação de performance. São três os tipos de testes de performance: TimedTest, LoadTest e ThreadedTest. Foi utilizado o TimedTest que executa um teste e mede o tempo transcorrido nesta ação.

A aplicação deu suporte as adaptações propostas, tendo gerenciado satisfatoriamente a recuperação dos objetos no SGBD, montagem do ambiente e respectivo envio ao *browser*. Os valores obtidos nessa avaliação são apresentados na Quadro 5.1.

Durante os testes, verificou-se que o tempo médio de resposta para construir um ambiente foi de 709,39 milisegundos (ms). Um ambiente que já foi visitado gasta em média 591,11 ms para ser carregado, e a atualização do ambiente devido à mudança de perfil possui tempo médio de 642,17 ms.

Quadro 5.1 Tempo de resposta do sistema para atualização do ambiente

Usuário	Nome do Ambiente	Tempo Gasto para Carregar Ambiente (ms)		
		Novo Ambiente	Ambiente Visitado Anteriormente	Ambiente com mudança de perfil
Marcus	Hall Entrada	747	632	678
Marcus	Queda Livre	712	581	673
Marcus	Pêndulo	705	579	672
Salerno	Hall Entrada	742	627	642
Salerno	Queda Livre	686	584	635
Salerno	Pêndulo	685	584	634
Daniel	Hall Entrada	747	629	638
Daniel	Queda Livre	703	578	642
Daniel	Pêndulo	700	579	641
Abella	Hall Entrada	745	621	646
Abella	Queda Livre	680	562	612
Abella	Pêndulo	685	564	610
Rodrigo	Hall Entrada	732	628	657
Rodrigo	Queda Livre	686	573	639
Rodrigo	Pêndulo	684	569	641
Fujioka	Hall Entrada	758	634	642
Fujioka	Queda Livre	688	557	628
Fujioka	Pêndulo	684	559	629
Tempo Médio		**709,39**	**591,11**	**642,17**

Os testes foram realizados para medir a performance do servidor. Conforme verificado no Quadro 5.1, considerando os resultados preliminares, pode-se concluir que a interface de comunicação cliente/servidor realizou a comunicação com o servidor de modo satisfatório, não havendo perda de informação nem queda na comunicação. Além disso, os ambientes foram carregados corretamente e atualizados conforme determinado pelos agentes. Neste caso, os testes realizados atenderam as expectativas projetadas para o VEPersonal, considerando que a infra-estrutura desenvolvida gerou e adaptou um AV em tempo real.

Os resultados obtidos numa rede local precisam ainda ser confrontados com os resultados de testes de avaliação de desempenho realizados em ambiente real de produção. Para tanto, é preciso hospedar o VEPersonal em um servidor Web para que o usuário possa acessá-lo utilizando a Internet, via protocolo TCP/IP. Contudo, os

servidores comerciais, com hospedagem paga, em sua maioria, ainda não utilizam SGBD com suporte a XML. A outra opção, os servidores de instituições governamentais, como por exemplo, os de Universidades, necessitam de autorização e de configuração específica ou de técnicos responsáveis para o suporte, o que dificulta sobremaneira a realização de testes desta natureza em tais ambientes.

Apesar do escopo deste trabalho ter sido o desenvolvimento de uma infra-estrutura para gerar ambientes adaptativos em função do perfil do usuário, tendo sido realizada a análise de desempenho para verificar a capacidade do VEPersonal em carregar novos ambientes solicitados pelo usuário e adaptar os já existentes, apresenta-se a seguir algumas orientações para a avaliação da interface do VEPersonal, tanto sob o ponto de vista de usabilidade como em relação à satisfação do usuário. Entende-se que a avaliação da interface deve ser realizada utilizando metodologias adequadas e, portanto, requer o envolvimento de uma equipe interdisciplinar para a sua validação e conseqüente consolidação dos resultados.

Atualmente, ainda existem poucos trabalhos sobre avaliação de interfaces em AV. Entretanto, algumas metodologias encontradas na literatura foram desenvolvidas para avaliar a usabilidade, propostas por Gabbard et al. (1999), Trevisan (2004) e Bastos (2007).

A avaliação da interface do VEPersonal pode ser realizada utilizando a proposta de Bastos (2007), cuja metodologia é aplicada para ambientes que já foram desenvolvidos. Essa metodologia é realizada em quatro passos:

1º) Avaliação heurística, em que especialistas em usabilidade avaliam o design da interface. Este tipo de avaliação deve ser realizado através de observações dos experimentos de Física desenvolvidos na aplicação e, em seguida, elaborado um documento com a opinião dos especialistas relatando sobre o que está adequado e o que está ruim na interface;

2º.) Análise da tarefa do usuário, através da observação das ações realizadas pelo usuário. Nsta análise, o usuário deve ter conhecimento prévio das tarefas a serem realizadas. A avaliação deve verificar se o usuário consegue interagir com a interface (e os objetos contidos nela), ou seja, executar os experimentos, navegar pelas salas e responder às perguntas. Esta etapa consiste em identificar as dificuldades encontradas pelo usuário;

3º) Avaliação Formativa é um método para julgar a importância de uma ação no ambiente. No VEPersonal, ela deve verificar as ações de interação do usuário no ambiente (clique nos botões de resposta, execução do experimento, alteração das propriedades do experiemento, consulta a novas informações, mudança de sala e solicitação de ajuda). Esta avaliação foi desenvolvida com o objetivo de coletar informações sobre as ações no ambiente com a intenção de melhorá-lo; e

4º) Avaliação sumativa tem o propósito de avaliar o impacto, a usabilidade e a efetividade do sistema, ou seja, de verificar se existe uma boa performance do usuário na interação com o ambiente. Bastante útil quando o projeto está completo. É nesta etapa que se identifica a eficácia do sistema, ou seja, sua capacidade de realizar o que foi projetado e se os objetivos foram atendidos.

Em relação à validação da satisfação do usuário deve-se, inicialmente, utilizar metodologias de aprendizagem adequadas, com avaliação do processo cognitivo necessário para a realização da adaptação. Tais metodologias devem ser desenvolvidas sob orientação de especialistas da área de educação e/ou do domínio do conhecimento modelado no ambiente virtual, o que foge ao escopo desta tese.

Entretanto, um possível roteiro de avaliação da satisfação do usuário para a aplicação de Experimentos em Física pode ser sugerido:

- Reunião com professores de Física do Segundo Grau e de Universidade para avaliação dos níveis de conhecimento aferidos pelo experimentos;
- Apresentação aos professores dos experimentos desenvolvidos (Queda Livre, Pêndulo Simples e Lançamento de Projéteis), além do Hall de Entrada que permite a entrada do usuário no sistema e a sua iniciação na busca de informações sobre o funcionamento dos experimentos;
- Aplicação de um questionário para ser respondido pelos professores para avaliação dos experimentos desenvolvidos, com o objetivo de identificar se o protótipo atende a finalidade de ensino e se é uma ferramenta importante para a melhoria do ensino de Física. Deve-se avaliar se a interface é de fácil entendimento e de fácil utilização;

- Este questionário também é útil para obter sugestões dos professores em relação à melhoria dos conteúdos dos experimentos e o aumento do grau de adaptatividade do sistema; e
- Uma vez feitas as inclusões das sugestões e a respectiva aprovação por parte dos professores, o protótipo deve ser disponibilizado aos usuários (alunos do 2º grau) para que eles possam avaliar a aplicação.

5.6 Considerações Finais

O desenvolvimento e a implementação do VEPersonal exigiu a utilização de diversas tecnologias, sendo necessário integrar componentes de realidade virtual com banco de dados, agentes e comunicação via Web.

Em relação ao armazenamento e gerenciamento dos objetos tridimensionais que compõem os AV, foi necessário criar mecanismos que atendessem aos requisitos das consultas ao SGBD PostgreSQL com suporte a XML, bem como que realizassem a extração do nível de informação de cada objeto, de acordo com o atributo *userLevel* definido no perfil do usuário. Para tanto, utilizou-se o padrão de projeto DAO (ver seção 5.2) que possibilita ao sistema gerar apenas as regras de negócio da consulta, abstraindo-se da complexidade de acesso ao Banco de Dados. Tais regras foram inseridas nas classes dos agentes, responsáveis por essa atividade. Uma vez recuperados os objetos, foi gerado um framework, utilizando o JDOM, para extrair os níveis adequados de cada objeto.

Em relação ao controle do ambiente (acompanhamento das ações do usuário e criação e atualização dos AV) foi necessário, inicialmente, criar uma ontologia (conceitos, ações e predicados) para que os agentes tivessem um entendimento comum sobre os objetos do domínio. Os diagramas de seqüência desenvolvidos no projeto do VEPersonal, auxiliaram no acompanhamento da troca de mensagens entre os agentes e na especificação de suas ações.

A interface de comunicação entre o sistema e o usuário, realizada via Web, utilizou diversos padrões de projeto, o que permitiu uma conexão cliente/servidor eficiente e sem perda de informação. O framework LipeRMI possibilitou manter a conexão do servidor com o usuário durante todo o período que ele interagiu com o

sistema e garantiu a transferência de objetos 3D e das ações realizadas pelo usuário no browser Xj3D.

A avaliação do desempenho do VEPersonal teve como objetivo verificar a integração de diversas tecnologias, determinando a viabilidade de utilizar agentes, SGBD XML e ambiente Web para o gerenciamento de AV adaptativos. Os resultados obtidos, ainda que iniciais, demonstram que o VEPersonal atende aos requisitos definidos no projeto (ver objetivos descritos no capítulo 1): gerar e manter AV em tempo real, via Web, utilizando objetos 3D através do reuso, com níveis diferentes de complexidade em função do perfil de cada usuário.

6. Conclusões e Trabalhos Futuros

A representação de ambientes virtuais com características próximas ao mundo real tem contribuído para o desenvolvimento de aplicações interativas e dinâmicas. Novos recursos têm sido adicionados para que os usuários possam navegar pelo ambiente, interagir com objetos, executar tarefas e responder a estímulos, com quase a mesma naturalidade que se estivessem executando tarefas em um mundo real. Em tais ambientes é possível, por exemplo, simular situações reais de perigo e treinar pessoas para atuarem em casos de emergência, ou ainda, gerar ambientes de jogos educativos em que os usuários interagem com o ambiente e entre si, e realizem atividades de forma lúdica e participativa.

Além disso, a adaptação de um AV em função do perfil do usuário pode aumentar o interesse e a participação do usuário no ambiente. Técnicas de adaptatividade foram propostas, gerando novas funcionalidades e proporcionando novos rumos para a personalização de ambientes virtuais. Essas técnicas visam gerar aplicações mais atrativas e, conseqüentemente, aumentar a produtividade do usuário. Por exemplo, lojas virtuais, museus temáticos e cidades em 3D, com visualização de informações direcionadas ao perfil do usuário, têm sido utilizadas em ambientes de navegação, tornando-os mais interativos e dinâmicos.

A evolução das linguagens de representação de objetos 3D, o aumento da capacidade de transmissão de dados pela Internet e a integração de tecnologias de comunicação, contribuíram para a expansão do acesso a ambientes virtuais na Web. Este acesso possibilitou aos usuários, localizados em qualquer lugar e com diferentes níveis de conhecimento, a interação com ambientes dinâmicos e interativos, através de conexões remotas.

Apesar desse avanço, verifica-se que nesses ambientes faltam mecanismos para identificar a evolução do conhecimento do usuário e propor atualizações, em tempo real, que satisfaçam as suas necessidades. Os ambientes que propõem algum tipo de adaptação de conteúdo fazem apenas uma reorganização dos objetos no ambiente, não aproveitando o potencial existente de modificação dos objetos e de outras mídias para

gerar conteúdos mais apropriados ao nível de conhecimento do usuário e/ou mais adequados ao seu perfil.

Neste trabalho de tese foi especificada, implementada e validada uma infra-estrutura para gerenciamento de componentes de AV adaptativos em tempo real, denominada VEPersonal. O sistema proposto visou solucionar o problema de adaptação de conteúdos com diversos níveis de detalhes que são apresentados à medida que o conhecimento do usuário aumenta. Esta técnica é importante, principalmente, para ambientes de ensino e de simulação em que existe um processo de aprendizagem durante a interação com o AV.

Os objetos utilizados na construção de AV foram armazenados em um SGBD com suporte a XML, sendo possível a reutilização desses objetos em outros ambientes. A utilização de agentes no gerenciamento de AV permitiu gerar as adaptações necessárias, de acordo com o perfil do usuário.

6.1 Soluções Adotadas para o VEPersonal

As propostas existentes para AV adaptativos, até o momento, não se preocupam com a adaptação de conteúdos ou com a modificação do ambiente à medida que o usuário aprende e se familiariza com o referido ambiente. A estrutura do VEPersonal foi desenvolvida considerando a necessidade de se realizar a geração e o monitoramento dos conteúdos de AV adaptativos, em tempo real.

Para atingir esse objetivo, algumas soluções foram utilizadas a fim de realizar a implantação do VEPersonal. As principais são:

- a linguagem X3D foi utilizada para representar ambientes virtuais, pois além de possuir recursos de visualização e representação de objetos 3D, ela possui estrutura XML, facilitando a manipulação e armazenamento de objetos, e a transferência de dados na Web;
- O uso de um SGBD com suporte a XML permitiu o armazenamento tanto de objetos 3D como de dados relacionais (perfil do usuário e estado do ambiente), garantindo persistência e integridade no processo de armazenamento e recuperação dos dados. O acesso através de uma interface de programação Java (JDBC) facilitou o

gerenciamento desses dados pelos agentes do sistema, que também são desenvolvidos em Java.

- a incorporação de um sistema multi-agentes permitiu a distribuição de tarefas para realização do acompanhamento das ações do usuário e para tomada de decisão durante a atualização dos AV. As consultas ao SGBD também foram realizadas pelos agentes, através de acessos simultâneos às informações. A troca de mensagens entre os agentes, bem como a comunicação com a interface, foi facilitada pelo uso do ambiente JADE;
- o Modelo do Usuário permitiu armazenar os perfis dos usuários, os seus dados pessoais e sua evolução cognitiva, bem como gerar um método para avaliação do seu perfil;
- a incorporação do Modelo do Ambiente contribuiu para manter o ambiente atualizado durante a interação do usuário e registrar as alterações ocorridas. O Modelo do Ambiente recupera o ambiente do usuário com todas as características de sua última visita. Desta forma, não há perda de conteúdo e é preservado o histórico da adaptatividade realizada durante a interação com o usuário. O Modelo do Ambiente pode também armazenar e gerenciar os diversos ambientes visitados pelos usuários que acessaram o sistema; e
- o desenvolvimento de uma interface de comunicação cliente/servidor contribuiu para gerar uma aplicação capaz de identificar as ações do usuário executadas no *browser* Xj3D, rodando na máquina do cliente, e enviá-las para a sociedade de agentes, executada no servidor. Este tipo de aplicação necessitou de soluções que integrassem as diversas tecnologias utilizadas. Para tanto, foi usado o *Java Web Start* que permitiu viabilizar o acesso ao *browser* Xj3D e protocolo LipeRMI para transferência de dados entre a interface e os agentes. Com esta integração, foi garantida a comunicação do servidor com a aplicação, possibilitando que os agentes gerenciassem uma aplicação remota sem perda de desempenho e de informação.

6.2 Contribuições

Considerando as características descritas acima, pode-se destacar as seguintes contribuições deste trabalho:

- **Detalhamento do conteúdo dos objetos em Ambientes Virtuais**

A geração de uma estrutura com níveis de detalhes em cada objeto, permitiu construir AV adaptativos com vários níveis de complexidade para um mesmo conteúdo. Estes níveis, representados pelo atributo *userLevel*, foram inseridos nos objetos através de *tags* XML. Os níveis são gerados em função dos perfis armazenados no Modelo do Usuário. A recuperação desses níveis é realizada a partir de uma consulta ao SGBD, sendo extraídos apenas os conteúdos que coincidem com o perfil do usuário. Isto permite gerar um ambiente diferente para cada usuário de acordo com seu perfil, sem precisar gerar um novo modelo do ambiente, bastando apenas modificar os níveis de complexidade dos respectivos conteúdos.

Neste caso, ao invés de gerar vários objetos com conteúdos semelhantes, é necessário apenas gerar um objeto com vários níveis de detalhes, evitando assim a replicação de objetos e a sobrecarga de armazenamento no SGBD.

O atributo *userLevel* também pode ser utilizado de uma forma mais ampla, podendo determinar o acesso do usuário a objetos e ambientes específicos em função de níveis de prioridade atribuídos a cada usuário. Desta forma, o usuário só teria acesso aos objetos (e ambientes) que tivessem o mesmo *userLevel* que ele.

- **Reuso de objetos na construção de Ambientes Virtuais**

A construção de um AV a partir da indexação de objetos armazenados em um SGBD com suporte a XML viabilizou a reutilização dos objetos na geração de novos ambientes. Os objetos armazenados no SGBD são instanciados a cada ambiente gerado, o que permite que tais objetos sejam reaproveitados em novos ambientes. Além disso, a estrutura de armazenamento de objetos 3D possibilitou a recuperação das informações com eficiência e rapidez.

- **Adaptatividade em tempo real**

A adaptação em tempo real foi alcançada com a agregação de inteligência ao VEPersonal, através de uma sociedade de agentes. Isto acelerou o processo de tomada de decisão, tanto para determinar os objetos a serem inseridos no Ambiente Virtual durante o processo de adaptação como para atualizar o perfil do usuário.

Além disso, a interface desenvolvida permitiu que a adaptação fosse realizada apenas com a inserção ou remoção de objetos no ambiente, sem a necessidade de carregar todo o AV novamente, agilizando o processo.

- **Gerenciamento do perfil do usuário e das atualizações do AV**

O processo de avaliação do aprendizado do usuário durante sua interação com o ambiente, utilizando um método associado a Fatores de Confiança para calcular o perfil, auxiliou os agentes na atualização do ambiente. O Modelo do Usuário, armazenado em um SGBD, permitiu maior eficiência na identificação dos perfis dos usuários que acessam o sistema, garantindo assim, a persistência dos dados.

O gerenciamento dos ambientes visitados pelos usuários e o histórico das adaptações realizadas durante o processo de interação, permite recuperar o último ambiente visitado pelo usuário com todas as modificações ocorridas e com os níveis de detalhes apresentados. Esta recuperação é importante no caso do retorno do usuário ao ambiente, em uma nova sessão ou devido à ocorrência de uma queda na conexão.

- **Desenvolvimento da arquitetura do VEPersonal e integração de tecnologias**

Uma importante contribuição desta tese é a arquitetura desenvolvida para o VEPersonal. Nesta proposta, foram utilizadas diversas tecnologias que integraram as áreas de Agentes Inteligentes, Banco de Dados e Interface Web com a de Realidade Virtual, o que possibilitou a construção de uma arquitetura robusta e inovadora. Os padrões de projeto contribuíram na padronização da interface de comunicação cliente/servidor, bem como no gerenciamento da transferência de dados entre os agentes e o SGBD. O fluxo de informação do sistema foi garantido através dos padrões Java, tendo sido utilizados diversos recursos desse software na solução dos problemas de integração.

As contribuições descritas nesta seção possibilitaram a criação e o gerenciamento de AV adaptativos de forma inovadora. Entretanto, algumas limitações podem ser encontradas, como por exemplo, o tratamento do acesso simultâneo e controle da conexão de vários usuários, a definição de uma ontologia para especificação de um meta-modelo de AV, e a elaboração de um sistema de ajuda ao usuário para orientação na execução de tarefas.

6.3 Trabalhos Futuros

Embora a especificação e a implementação do VEPersonal apresentarem resultados positivos e satisfatórios em relação ao gerenciamento de ambientes virtuais adaptativos, alguns trabalhos podem ser explorados, possibilitando a geração de novas pesquisas e o aumento da potencialidade do VEPersonal. Sugere-se como trabalhos futuros:

- Gerenciamento do acesso simultâneo de vários usuários ao VEPersonal, bem como o acompanhamento das suas ações. Os protocolos de comunicação dos agentes deverão incluir a identificação do usuário e a conexão com o SGBD (para cada usuário existe uma conexão), o que facilitará a recuperação do ambiente para cada usuário e o respectivo processo de atualização. Isto permitirá que vários usuários conectados remotamente, com diferentes perfis e diferentes prioridades, acessem o sistema ao mesmo tempo (a ser realizado em Trabalho de Graduação ou em projeto de Iniciação Científica);

- Utilização de técnicas de Aprendizagem de Máquina, principalmente a abordagem colaborativa, para a modelagem do usuário. Estas técnicas poderiam auxiliar na previsão sobre os padrões de comportamento e/ou de conhecimento de um usuário individual em relação a um grupo de usuários, contribuindo na identificação e atualização do seu perfil. Desta forma, o perfil inicial do usuário poderia ser determinado com maior precisão, gerando assim um AV mais adaptado às necessidades do usuário (a ser realizado em uma dissertação de Mestrado na área de Inteligência Artificial);

- Mudança do mecanismo de coordenação da estrutura mestre/escravo para P2P, com o objetivo de aumentar a autonomia de atuação dos agentes. Neste caso, cada agente seria responsável por um conjunto de tarefas e as executaria em função da negociação com os outros agentes. Este mecanismo poderia ser bastante útil no gerenciamento de aplicações multi-ambientes e, conseqüentemente, aumentaria o poder de geração de ambientes virtuais adaptativos (a ser realizado em Trabalho de Graduação);

- Utilização de um *chatterbot* (assistente virtual capaz de conversar com o usuário) para responder às perguntas do usuário, possibilitando o esclarecimento de dúvidas e solicitação de serviços (a ser realizado em projeto de Iniciação Científica e Trabalho de Graduação);

- Uso de uma ontologia para descrever um conjunto de metadados dos objetos, sendo utilizada para especificar as entidades existentes na representação dos AV. A ontologia pode auxiliar na automatização da construção de ambientes virtuais, na decomposição e armazenamento de ambientes já existentes, e na representação de níveis de complexidade, evitando a geração de objetos mal-formados (a ser realizado em uma dissertação de Mestrado);

- Utilização de um SGBD com suporte a XQuery [XQuery, 2007]. XQuery foi recomendada pela W3C em janeiro de 2007 e tem a vantagem de fazer junções entre dados XML e obter resultados com uma estrutura diferente da existente no documento. A aplicação de XQuery permite recuperar, em uma única consulta, os objetos 3D com as *tags* XML correspondentes ao nível de detalhes indicado pelo *userLevel*, agilizando o processo de recuperação das informações no SGBD e conseqüente diminuição no tempo de resposta de atualização de um AV (a ser realizado em Trabalho de Graduação);

- Utilização da arquitetura do VEPersonal para outras aplicações que tenham necessidade de acompanhamento da evolução de aprendizagem, como por exemplo, jogos educativos ou ambientes de simulação. Em particular, ambientes de simulação com permissões diferenciadas para execução de tarefas possibilitariam realizar uma melhor avaliação do desempenho do sistema e aprimorar o processo de decisão dos agentes, tendo em vista que este tipo de aplicação está mais próximo de atividades executadas em tempo real (a ser realizado em uma dissertação de Mestrado); e

- Desenvolvimento de ferramentas de autoria para o VEPersonal com o objetivo de gerar os ambientes virtuais e armazená-los no SGBD, além de gerar as regras de negócios para cada ambiente (a ser realizado em uma dissertação de Mestrado e Trabalhos de graduação associados).

Referências Bibliográficas

[Abbattista et al., 2002] Abbattista, F.; Degemmis, M; Fanizzi, N.; Licchelli, O. Lops, P.; Semeraro, G.; Zambetta, F. *Learning User Profile for Content-Based Filtering in e-Commerce*. Workshop Apprendimento Automatico: Metodi e Applicazioni, Siena, 2002.

[ActiveWorlds, 2007] ActiveWorlds Corporation. http://www.activeworlds.com. Último acesso em abril/2007.

[ActiveX, 2007] ActiveX Controls. http://msdn.microsoft.com/library/default.asp?url=/workshop/components/activex/activex_node_entry.asp. Último acesso em abril de 2007.

[Adabas, 2007] Adabas 2006 Portfolio Overview. http://www1.softwareag.com/Corporate/products/adabas/portfolio/default.asp. Último acesso em abril de 2007.

[Albuquerque e Almeida, 2006] Albuquerque, M. V. A; Almeida, R. M. G. *Escolha entre Banco de Dados Relacional e Banco de Dados XML: Estudo de Caso para o Projeto VEPersonal*. (Trabalho de Graduação). UNIPE - Centro Universitário de João Pessoa. Curso de Bacharelado em Ciência da Computação. João Pessoa-PB, 2006.

[Alpert et al., 2003] Alpert, S. R.; Karat, J.; Karat, C-M.; Brodie, C.; Vergo, J. G. *User attitudes regarding a user-adaptive ecommerce web site*. User Modeling and User-Adapted Interaction. 2003.

[Amazon, 2007] Amazon.com. http://www.amazon.com.. Último acesso em abril de 2007.

[Ames et al., 1997] Ames, L. A.; Nadeau, D. R.; Moreland, J. L. *The VRML 2.0 Sourcebook*. 2nd. Edition, John Wiley & Sons, Inc., 1997.

[Anastassakis et al., 2001] Anastassakis, G.; Ritching, T.; Panayiotopoulos, T. *Multi-agent Systems as Intelligent Virtual Environments*. LNAI 2174, pp. 381-365, 2001.

[Anjaneyulu, 1997] Anjaneyulu, K. S. R.,. *Concept Level Modeling on the WWW*. In: Brusilovsky, P., Nakabayashi, K. and Ritter, S. (eds.) Proc. of Workshop Intelligent Educational Systems on the World Wide Web at AI-ED'97, 8th World Conference of the Artificial Intelligence in Education, Kobe, Japan, ISIR, pp. 26-29, 1997.

[Apache.org, 2007] The Apache Software Foundation. http://www.apache.org. Último acesso em setembro de 2007.

[Aquino et al., 2005a] Aquino, M. S.; Souza, F. F.; Frery, A. C. *A Multi-Agent Architecture for Generating and Monitoring Adaptive Virtual Environments*. 5th International Conference on Hybrid Intelligent Systems – HIS'05, Rio de Janeiro-RJ, Brazil, November 06-09, p.515-517. Publisher: IEEE Computer Society, Washington, DC, USA, 2005.

[Aquino et al., 2005b] Aquino, M. S.; Souza, F. F.; Frery, A. C. *VEPersonal – An infrastructure of Virtual Reality Components to Generate Web Adaptive Environments*. ACM International Conference Proceeding Series; Vol. 125. Proceedings of the 11th Brazilian Symposium on Multimedia and the Web – WebMedia 2005, Poços de Caldas-MG, Brazil, December 05 – 07, pp. 1-8, 2005.

[Aquino et al., 2006] Aquino, M. S.; Souza, F. F.; Frery, A. C.; Neto, L. G. A.; Albuquerque, M V. A.; Almeida, R. M. G. *Adaptação de Conteúdos pelo Perfil do Usuário para Personalização de Ambientes Virtuais com X3D*. VIII Symposium on Virtual Reality, SVR 2006, Belém-PA, Brazil, May 02-05, 2006.

[Aquino et al., 2007] Aquino, M. S.; Souza, F. F.; Frery, A. C; Souza, D. A. C. M.; Fujioka, R. C. *Supporting Adaptive Virtual Environments with Intelligent Agents*. 7th International Conference on Intelligent Systems Design and Applications, ISDA'07, Publisher: IEEE Computer Society, Washington, DC, USA. Rio de Janeiro-RJ, Brazil, October, 22-24, pp. 217-222, 2007.

[Ardissono, 1999] Ardissono, L.; Goy, A. *Tailoring the interaction with users in electronic shops*. In: Proc. of UM99. 7th International Conference on User Modeling. Springer Verlag, pp. 35-44, 1999.

[Ballegooij e Eliëns, 2001] Ballegooij, A.; Eliëns, A. *Navigation by Query in Virtual Worlds*. Proceedings of the 6th International Conference on 3D Web Technology, Germany, February, 2001.

[Bastos, 2007] Bastos, N. C. *Uma Metodologia para Avaliação de Usabilidade de Interfaces de Realidade Mista Interativas*. (Dissertação de Mestrado) Recife: Pós-Graduação em Ciência da Computação do CIn/UFPE, 2007.

[Beck, 2000] Beck, K. *Extreme Programing Explained: Embrace Change*. Addison-Wesley, 2000.

[Bellifemine et al., 2001] Bellifemine, F.; Poggi, A.; Rimassa, G. *Developing multi-agent systems with JADE*. Lecture Notes in Computer Science: Intelligent Agents VII. Seventh International Workshop on Agent Theories, Architectures, and Languages (ATAL-2000), Boston, MA., 2001.

[Bellifemine et al., 2003] Bellifemine, F.; Caire, G.; Poggi, A.; Rimassa, G. JADE - A White Paper. In:Telecom Italia EXP magazine, Volume 3, n. 3, September 2003. http://jade.tilab.com/papers/2003/WhitePaperJADEEXP.pdf. Último acesso em abril de 2007.

[BerkeleyDB, 2006] *A Comparison of Oracle Berkeley DB and Relational Database Management Systems*. An Oracle Technical White Paper. 2006. http://www.oracle.com/database/docs/Berkeley-DB-v-Relational.pdf. Último acesso em abril de 2007.

[BerkeleyDB, 2007a] Oracle Berkeley DB Product Family. http://www.oracle.com/database/berkeley-db/index.html. Último acesso em abril de 2007.

[BerkeleyDB, 2007b] Oracle Berkeley DB XML. http://www.oracle.com/technology/products/berkeley-db/xml/index.html. Último acesso em abril de 2007.

[Boticario et al., 2005] Boticário, J.; Santos, O.; Van Rosmalen, P. I*ssues in developing standard-based adaptive learning management systems*. EADTU 2005 Working Conference: Towards Lisbon 2010: Collaboration for Innovative Content in Lifelong Open and Flexible Laerning, 2005.

[Bourret, 2005] Bourret, R. *XML Database Products*. http://www.rpbourret.com/xml/XMLAndDatabases.htm. Último acesso em abril de 2007.

[Browne, 2004] BROWNE, C. *SQL Databases*, 2004 http://cbbrowne.com/info/rdbmssql.html. Último acesso em abril de 2007.

[Carberry, 2001] Carberry, S. *Techniques for Plan Recognition*. User Modeling and User-Adapted Interaction, vol. 11, Kluwer Academic Publischers, pp. 31-48, 2001.

[Castellano et al., 2001] Castellano, G.; Fanelli, A. M.; Roselli, T. *Mining Categories of Learns by a Competitive Neural Network*. Proceedings of the INNS – IEEE International Joint Conference on Neural Networks, Washington, July, 2001.

[Cavazza et al., 2001] Cavazza, M.; Charles, F.; Mead, S.; Strachan, A. *Virtual Actors' Behaviour for 3D Interactive Storytelling*. Proceedings of the Eurographics Conference, 2001

[Celentano et al., 2004] Celentano, A.; Nodari, M.; Pittarello, F. *Adaptive Interaction in Web3D Virtual Worlds*. Web3D 2004, 9th International Conference on 3DWeb Technology, Monterey, California, USA, April 5-8, 2004.

[Chittaro e Ranon, 2000a] Chittaro L.; Ranon R. *Adding Adaptive Features to Virtual Reality Interfaces for E-Commerce*. Proceedings of AH-2000: International Conference on Adaptive Hypermedia and Adaptive Web-based Systems, Lecture Notes in Computer Science 1892, Springer-Verlag, Berlin, pp. 86-97, 2000.

[Chittaro e Ranon, 2000b] Chittaro, L.; Ranon, R. *Virtual Reality stores for 1-to-1 E-commerce*. Proceedings of CHI 2000 Workshop on Designing Interactive Systems for 1-to-1 E-Commerce, The Hague, The Netherlands, 2000.

[Chittaro e Ranon, 2002a] Chittaro, L.; Ranon, R. *New Directions for the Design of Virtual Reality Interfaces to E-Commerce Sites*. Proceedings of AVI 2002: 5th International Conference on Advanced Visual Interfaces, ACM Press, New York, pp. 308-315, May 2002.

[Chittaro e Ranon, 2002b] Chittaro, L.; Ranon, R. *Dynamic generation of personalized VRML content: a General Approach and its Application to 3D E-Commerce*. Proceedings of Web3D 2002: 7th

International Conference on 3D Web Technology, ACM Press, New York, pp.145–154, February 2002.

[Chittaro et al., 2003] Chittaro, R.; Ranon, R.; Ieronutti, L. *Guiding Visitors of Web3D Worlds through Automatically Generated Tours*. Proceedings of Web3D 2003: 8th International Conference on 3D Web Technology, ACM Press, New York, pp. 27-38, March 2003.

[Conrad, 2004] Conrad, T. *PostgreSQL vs. MySQL vs. Commercial Databases: It's All About What You Need*. DevX.com. http://www.devx.com/dbzone/Article/20743. Último acesso em abril de 2007.

[DAO, 2007] Core J2EE Patterns - Data Access Object. http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html. Último acesso em abril de 2007.

[DB2, 2007] DB2 Universal Database. http://www-306.ibm.com/software/br/db2/data/db2universal_db.shtml. Último acesso em abril de 2007.

[De Antonio et al., 2005] De Antonio, A., Ramírez, J., Imbert, R., Méndez, G.; Aguilar, R. *A Software Architecture for Intelligent Virtual Environments Applied to Education*. Revista de la Facultad de Ingeniería, Univ. de Tarapacá. Vol. 13 No. 1, Arica: Chile. pp. 47-55, 2005.

[Devillers et al., 2002] Devillers F.; Donikian S.; Lamarche F.; Taille J.F. A *Programming Environment for Behavioural Animation*. Journal of Visualization and Computer Animation; 13: pp. 263-274, 2002. http://www.irisa.fr/siames/Fabrice.Lamarche/Articles/JVCA02.pdf. Último acesso em abril de 2007.

[DirectX, 2007] Microsoft DirectX. http://www.microsoft.com/windows/directx/default.mspx. Último acesso em abril de 2007.

[Distani et al., 2005] Dastani, M; Jacob N; Jonker, C. M.; Treur, J. *Modelling user preferences and mediating agents in electronic commerce*. Knowledge-Based Systems, Vol. 18, Issue 7, November 2005, pp. 335-352.

[DLL, 2007] Dynamic-link library – DLL. http://msdn2.microsoft.com/en-us/library/ms682589.aspx. Último acesso em setembro de 2007.

[DOM, 2007] Document Object Model (DOM). http://www.w3.org/DOM. Último acesso em setembro de 2007.

[DTD, 2007] XML specification DTD. http://www.w3.org/XML/1998/06/xmlspec-v21.dtd. Último acesso em abril de 2007.

[Eclipse, 2007] Eclipse - an open development platform. http://www.eclipse.org. Último acesso em abril de 2007.

[Eiter e Mascardi, 2002] Eiter, T.; Mascardi, V. *Comparing environments for developing software agents*. AI Communications Journal,Volume 15 , Issue 4. pp. 169-197, 2002.

[Estadao, 2007] *Os melhores lugares de Second Life*. Seção Link do Jornal O Estado de São Paulo. http://www.link.estadao.com.br/index.cfm?id_conteudo=10333. Último acesso em setembro de 2007.

[Estalayo et al., 2004] Estalayo, E.; Salgado, L.; Morán F.; Cabrera, J. *Applications Adapting Multimedia Information Association in VRML Scenes for E-Learning*. In: Proceedings of the 1st International Workshop LET-WEB3d'04, pp. 16-22, 2004.

[FIPA, 2007] Foundation for Intelligent Physical Agents – FIPA. http://www.fipa.org. Último acesso em abril de 2007.

[FIPA-ACL, 2007] FIPA Agent Communication Language Specifications. http://www.fipa.org/repository/aclspecs.html. Último acesso em abril de 2007.

[Fischer, 2001] Fischer, G. *User Modeling in Human-Computer Interaction*. Contribution to the 10th Anniversary Issue of the Journal "User Modeling and User-Adapted Interaction", vol. 11, no. 1/2, 2001.

[Frery et al., 2002] Frery, A.; Kelner, J.; Moreira, J., Teichrieb, V. *Satisfaction through Empathy and Orientation in 3D Worlds*. CyberPsychology and Behavior, 5(5): 451-459, 2002.

[Gabbard et al., 1999] Gabbard J.L., Hix, D. Swan, J.E. *User-centered design and evaluation of virtual environments.* IEEE Computer Graphics and Applications. Loa Alamitos , CA, USA. IEEE Computer Society Press, v.19, n.6, pp. 51-59, 1999.

[Gama et al., 2002] Gamma, E.; Helm, R.; Johnson, R.; Vlissides J. *Design patterns: abstraction and reuse of object-oriented design*, pp. 701 - 717 Springer-Verlag New York, Inc., 2002.

[GeoVRML, 2007] GeoVRML.org. http://www.ai.sri.com/geovrml. Último acesso em abril de 2007.

[Giarratano e Riley, 2004] Giarratano, J.; Riley, G. *Expert Systems: Principles and Programming*. 4th ed., PWS Publishing Company. International Thomson Publishing Company, 2004.

[Grand e Cliff, 1998] Grand, S.; Cliff, D. *Creatures: Entertainment software agents with artificial life*. Autonomous Agents and Multi-Agent Systems, v. 1, n. 1, p. 39-57. 1998.

[Gratch et al., 2002] Gratch, J.; Rickel, J.; Andre, E.; Badler, N.; Cassell, J.; Petajan, E. *Creating Interactive Virtual Humans: Some Assembly Required*. IEEE Intelligent Systems, v. 17, n. 4 (Special issue on AI in Interactive Entertainment), 2002.

[Herlocker, 2004] Herlocker, J. L.; Konstan, J. A., Terveen, L. G.; Riedl J. T. *Evaluating collaborative filtering recommender systems*. ACM Transactions on Information Systems, Vol. 22, No. 1, 2004.

[Höppner, 2003] Höppner, S. *An Agents' Definition Framework and a Methodology for Deriving Agents' Taxonomies*. KI 2003: Advances in Artificial Intelligence Volume 2821/2003. Publisher Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2003.

[Huang et al., 2000] Huang, Z.; Eliëns, A.; van Ballegooij, A. De Bra, P. *A Taxonomy of Web Agents*, IEEE Proceedings of the First International Workshop on Web Agent Systems and Applications (WASA 2000), 2000.

[Huang et al., 2001] Huang, Z.; Eliëns, A.; De Bra, P. *An Architecture for Web Agents*. Proceedings of the Conference EUROMEDIA'2001, SCS, 2001.

[Huang et al., 2002] Huang, Z.; Eliëns, A.; Visser, C. *3D Agent-based Virtual Communities*. Proceedings of the 3D Technologies for the World Wide Web, pp 137-143, ACM Press, 2002.

[JADE, 2007] Java Agent DEvelopment framework – JADE. http://jade.tilab.com. Último acesso em abril de 2007.

[Jameson e Wittig, 2001] Jameson, A.; Wittig, F. *Leveraging Data About Users in General in the Learning of Individual User Models*. In: Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence. Ed. B. Nebel. pp.1185–1192. San Francisco, CA: Morgan Kaufman, 2001.

[Jameson, 1999] Jameson, *A. User-adaptive systems: An integrative overview tutorial*. 7th International Conference on User Modeling - UM99, 1999; e 16th International Joint Conference on Artificial Intelligence - IJCAI99, 1999. http://dfki.de/~jameson/um99-tutorial-index.html. Último acesso em abril de 2007.

[Java, 2007] About Java Technology. http://www.sun.com/java/about. Último acesso em abril de 2007.

[Java3D, 2007a] Java 3D API. http://java.sun.com/developer/onlineTraining/java3d/. Último acesso em abril de 2007.

[Java3D, 2007b] Java 3D API Tutorial. http://java.sun.com/developer/onlineTraining/java3d. Último acesso em abril de 2007.

[JavaScript, 2007] Core JavaScript Guide http://devedge-temp.mozilla.org/library/manuals/2000/javascript/1.5/guide. Último acesso em abril de 2007.

[JDBC, 2007] Java Database Connectivity - JDBC. http://java.sun.com/javase/technologies/database. Último acesso em abril de 2007.

[JDK, 2007] Java SE Development Kit. http://java.sun.com/javase/downloads/index.jsp. Último acesso em abril de 2007.

[JDOM, 2007] JDOM Project. http://www.jdom.org. Último acesso em abril de 2007.

[Jennings, 2000] Jennings, N. R. *On Agent-based Software Engineering*. Artificial Inteligence, 117(2), pp. 277-296, 2000.

[JMeter, 2007] Apache JMeter. http://jakarta.apache.org/jmeter. Último acesso em abril de 2007.

[Joachims et al., 2005] Joachims, T.; Granka L.; Pan B.; Hembrooke H.; Gay, G. *Accurately interpreting clickthrough data as implicit feedback*. In Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR), pp. 154–161, 2005.

[Joachims et al., 2007] Joachims, T.; Granka L.; Pan B.; Hembrooke H.; Radlinski F.; Gay, G. *Evaluating the accuracy of implicit feedback from clicks and query reformulations in Web search*. In: Proceedings of the ACM Transactions on Information Systems (TOIS), Volume 25 Issue 2. 2007.

[Joerding, 1997] Joerding, T. *User Modeling for Electronic Catalogs and Shopping Malls in the World Wide Web*. In: Proc. of the 1st Workshop on Adaptive Systems and User Modeling on the WWW, 1997.

[Joerding, 1999] Joerding, T. *A Temporary User Modeling Approach for Adaptive Shopping on the Web*. In: Proc. of the 2nd Workshop on Adaptive Systems and User Modeling on the WWW, 1997.

[Johnson et al., 1998] Johnson, W. L.; Rickel, J.; Stiles, R.; Munrol, A. *Integrating Pedagogical Agents into Virtual Environments*. Presence: Teleoperators and Virtual Environments 7(6): 523-546, December 1998.

[Johnson, 2001] Johnson, W. L. *Pedagogical agent research at CARTE*. AI Magazine, 22: 85-94. Winter, 2001.

[JScript, 2007] JScript - Windows Script Technologies. http://msdn2.microsoft.com/en-us/library/hbxc2t98.aspx. Último acesso em abril de 2007.

[JUnitPerf, 2006] *In pursuit of code quality: Performance testing with JUnitPerf*, by Andrew Glover. http://www-128.ibm.com/developerworks/java/library/j-cq11296.html. Último acesso em setembro de 2007.

[JVM, 2007] Java Virtual Machine. http://en.wikipedia.org/wiki/Java_Virtual_Machine. Último acesso em abril de 2007.

[JWS, 2007] Java Web Start Technology. Sun Developer Network. http://java.sun.com/products/javawebstart. Último acesso em abril de 2007.

[Kass e Finin, 1988] Kass, R.; Finin, T. *Modeling the User in Natural Language Systems*. Computational Linguistics, vol. 14, no. 3, pp. 5-22, 1988.

[Kerberos, 2007] Kerberos: *The Network Authentication Protocol*. http://web.mit.edu/kerberos/www. Último acesso em abril de 2007.

[Kernighan e Ritchie, 1988] Kernighan, B. W.; Ritchie, D. M. *The C Programming Language*. Second Edition. Prentice Hall, 274 pp, 1988.

[Khatri et al., 2006] Khatri, V. S.; Sokolof, N.; Dadarkar, M. *Migrate from MySQL or PostgreSQL to DB2 Express-C*. IBM developerWorks, 2006. http://www-128.ibm.com/developerworks/db2/library/techarticle/dm-0606khatri.

[Kim et al., 2006] Kim, S.; Lee, K.; Hong, T.; Jung, M. *Modeling in Multi-Resolution and Its Applications*. Journal of Computer Science and Technology, Vol. 21, No.2, Publisher Springer Boston, pp.272-278, 2006.

[Kirner e Siscoutto, 2007] Kirner, C.; Siscoutto, R. *Fundamentos de Realidade Virtual e Aumentada*. In: Realidade Virtual e Aumentada: Conceitos, Projeto e Aplicações. Eds. Cláudio Kirner e Robson Siscoutto. Livro do Pré-Simpósio, IX Symposium on Virtual and Augmented Reality, Petrópolis – RJ, 2007.

[Kirner et al., 2004] Kirner, C.; Kirner T. G.; Júnior, N. C.; Buk C. V. *Uso de Realidade Aumentada em Ambientes Virtuais de Visualização de Dados*. In: Proceedings of VII Symposium on Virtual Reality – SVR, São Paulo - SP, outubro, 2004.

[Kirner e Tori, 2006] Kirner, C.; Tori, R. *Fundamentos de Realidade Aumentada*. In: Fundamentos e Tecnologia de Realidade Virtual e Aumentada. Eds. Romero Tori, Cláudio Kirner e Robson Siscoutto. Livro do Pré-Simpósio, VIII Symposium on Virtual and Augmented Reality, Belém – PA, 2006.

[Koychev e Schwab, 2000] Koychev, I.; Schwab, I. *Adaptation to drifting user's interests*. In Proceedings of ECML 2000 Workshop: Machine Learning in New Information Age. Barcelona, Spain, 2000.

[Lesser, 1999] Lesser, V. R. *Cooperative Multiagent Systems: a Personal View of the State of the Art*. Knowledge and Data Engineering, IEEE Transactions, 11(1), pp. 133-142, 1999.

[Levy e Weld, 2000] Levy, A.; Weld, D. *Intelligent Internet Systems*. Artificial Intelligence, vol. 118, no. 1-2, pp. 1-14, 2000.

[Lieberman, 1995] Lieberman, H. *Letizia: An Agent That Assist Web Browsing*. International Joint Conference on Artificial Intelligence, Montreal, pp. 924-929, 1995.

[Linden, 2007] Linden Lab – *Where Worlds are Born*. http://lindenlab.com. Último acesso em setembro de 2007.

[Linux, 2007] The Linux Kernel Archives. http://www.kernel.org. Último acesso em abril de 2007.

[LipeRMI, 2007] Lipe RMI – A Light Weight Internet Approach for Remote Method Invocation. http://lipermi.sourceforge.net. Último acesso em setembro de 2007.

[Liquibase, 2007] Database Refactoring With LiquiBase. http://www.liquibase.org. Último acesso em setembro de 2007.

[LSL, 2007] Linden Scripting Language - LSL. http://secondlife.com/whatis/scripting.php. Último acesso em setembro de 2007.

[Lu et al., 2005] Lu, H.; Xu, Y. J.; Wang, G.; Zheng, S.; Jiang, H.; Yu, G.; Zhou, A. *What makes the differences: benchmarking XML database implementations*. ACM Transactions Internet Techniques. 5(1): pp. 154-194. 2005.

[Martins, 2007] Martins, N. M. P. *Integração com Java – Scene Access Interface*. http://www.sal.ipg.pt/user/estg/martins/x3d.htm. Último acesso em abril de 2007.

[Matsuba, 2006] Matsuba, S. N. *Interview Yumetech – Xj3D*. 3d-test – 3D temps réel et interactive. http://www.3d-test.com/interviews/xj3d_1.htm. Último acesso em abril de 2007.

[Mattar e Maia, 2007] Mattar, J.; Maia, C. *Second Life da EaD & Vida Nova para o Professor Virtual: Caixa de Ferramentas 2.0 para o Aututor*. 13º Congresso da Associação Brasileira de Educação a Distância – ABED, Curitiba - PR, 2007. http://www.abed.org.br/congresso2007/tc/528200722418PM.pdf. Último acesso em outubro/2007.

[Mello, 2002] Mello, R. S. *Uma Abordagem Bottom-Up para a Integracao Semantica de Esquemas XM*L, (Tese de doutorado), Universidade Federal do Rio Grande do Sul. Porto Alegre, RS, Brasil, 2002.

[MPEG, 2007] Moving Picture Experts Group - MPEG. http://www.mpeg.org/MPEG/index.html. Último acesso em abril de 2007.

[MSOffice, 2007] Welcome to Microsoft® Agent. http://www.microsoft.com/msagent/default.asp. Último acesso em abril de 2007.

[Musse, 2000] Musse, S. *Human Crowd Modelling with Various Levels of Behaviour Control* (Phd Thesis). École Polytechnique Fédérale de Lausanne – EPFL, Lausanne, Suisse, 2000.

[Nijholt e Hulstijn, 2000] Nijholt, A.; Hulstijn, J. *Multimodal Interactions with Agents in Virtual Worlds*. In: Kasabov, N. (ed.): Chapter 8 - Future Directions for Intelligent Information Systems and Information Science, Physica-Verlag: Studies in Fuzziness and Soft Computing, pp. 148-173, 2000.

[Nikolopoulos, 1997] Nikolopoulos, *C. Expert Systems – Introduction to First and Second Generation and Hybrid Knowledge Based Systems*. Eds.: Marcel Dekker, New York, 1997.

[ODBC, 2007] ODBC – Open Database Connectivity Overview - http://support.microsoft.com/kb/110093/en-us. Último acesso em abril de 2007.

[Open Inventor, 2007] Open Inventor. http://oss.sgi.com/projects/inventor. Último acesso em setembro de 2007.

[OpenGL, 2007] The Industry's Foundation for High Performance Graphics. http://www.opengl.org. Último acesso em abril de 2007.

[OpenSceneGraph, 2007] OpenSceneGraph - Introduction. http://www.openscenegraph.org/projects/osg/wiki/About/Introduction. Último acesso em setembro de 2007.

[Oracle, 2007a] Oracle Corporation. http://www.oracle.com. Último acesso em abril de 2007.

[Oracle, 2007b] Oracle XML DB. http://www.oracle.com/technology/tech/xml/xmldb/index.html. Último acesso em abril de 2007.

[Oracle, 2007c] Oracle® XML DB Developer's Guide10g Release 2 (10.2) Part Number B14259-02 http://download-east.oracle.com/docs/cd/B19306_01/appdev.102/b14259/toc.htm. Último acesso em abril de 2007.

[Oracle10g, 2007] Oracle Database 10g http://www.oracle.com/technology/products/database/oracle10g/index.html. Último acesso em abril de 2007.

[Oracle9i, 2007] Oracle9i Database http://www.oracle.com/technology/software/products/oracle9i/index.html. Último acesso em abril de 2007.

[Paliouras et al., 1999] Paliouras, G.; Karkaletsis, V.; Papatheodorou, C.; Spyropoulos, C. *Exploiting Learning Techniques for the Acquisition of User Stereotypes and Communities*. Proceedings of the 7th International Conference on User Modeling, Canada, June, 1999.

[Panayiotopoulos et al., 1999] Panayiotopoulos, T.; Zacharis, N.; Vosinakis, S. *Intelligent Guidance in a Virtual University*. Advances in Intelligent Systems – Concepts, Tools and Applications, pp. 33-42, Kluwer Academic Press, 1999.

[Papatheodorou, 2001] Papatheodorou, C. *Machine Learning in User Modeling*. Machine Learning and Applications. Lecture Notes in Artificial Intelligence. Springer Verlag, 2001.

[Pires et al., 2006] Pires, C. E.; Nascimento, R. O.; Salgado, A. C. *Comparativo de Desempenho entre Bancos de Dados de Código Aberto*. In: Escola Regional de Banco de Dados, 2006, Passo Fundo - RS. Anais da ERBD06. Porto Alegre : SBC, p. 21-26, 2006.

[Pitarello e Celentano, 2001] Pittarello, F.; Celentano, A. *Interaction lócus – a multimodal approach for the structuring of virtual spaces*. In Proceedings of HCItaly 2001 Symposium, 2001.

[Pitarello, 2001] Pittarello, F. *Multi sensory 3d tours for cultural heritage: the palazzo grassi experience*. In Proceedings of ICHIM2001, Cultural heritage and technologies in the 3rd millennium, 2001.

[Pogi, 2007] Poggi, A. *Multi-agent System – JADE* (Notas de Aula). http://aot.ce.unipr.it/team/poggi/teaching/ia/docs/JADE.pdf. Último acesso em abril de 2007.

[Pohl, 1999] Pohl, W. *Logic Based Representation and Reasoning for User Modeling Shell Systems*. User Modeling and User Adapted Interaction V. 9, n. 3, pp. 217-282, 1999.

[Posix, 2007] IEEE POSIX® Certification Authority. http://standards.ieee.org/regauth/posix/index.html. Último acesso em abril de 2007.

[PostgreSQL, 2006] Perguntas Freqüentes (FAQ) sobre PostgreSQL. http://www.postgresql.org/docs/faqs.FAQ_brazilian.html#item1.1. Último acesso em abril de 2007.

[PostgreSQL, 2007a] PostgreSQL 8.2 Released. http://www.postgresql.com. Último acesso em abril de 2007.

[PowerDesigner, 2007] PowerDesigner – *Sybase's all-in-one enterprise modeling and design solution*. http://www.sybase.com/products/modelingmetadata/powerdesigner. Último acesso em setembro de 2007.

[Proxool, 2007] Proxool 0.9.0RC3. http://proxool.sourceforge.net. Último acesso em setembro de 2007.

[Reynolds, 2001] Reynolds, C. W. *Interaction with Groups of Autonomous Characters*. In: Proceedings of Game Developers Conference 2000, CMP Game Media Group, San Francisco, CA, pp. 449-460, 2001.

[Rickel e Johnson, 1997] Rickel, J.; Johnson, W. *Integrating Pedagogical Capabilities in a Virtual Environment Agent*. Proceedings of the 1st International Conference on Autonomous Agents, February, ACM Press, 1997.

[Rickel e Johnson, 2000] Rickel, J.; Johnson, W. *Task-Oriented Collaboration with Embodied Agents in Virtual Worlds*. Embodied Conversational Agents, MIT Press, Boston, 2000.

[Rickel et al., 2002] Rickel, J.; Marsella, S.; Gratch, J.; Hill, R.; Traum, D.; Swartout W. *Toward a New Generation of Virtual Humans for Interactive Experiences*. IEEE Intelligent Systems, vol. 17, no. 4, Special issue on AI in Interactive Entertainment, 2002.

[Rizzo et al., 2002] Rizzo, A. A.; Bowerly, T.; Buckwalter, J. G.; Schultheis, M.; Matheis, R.; Shahabi, C.; Neumann, U.; Kim, L.; Sharifzadeh, M. *Virtual Environments for the Assessment of Attention and Memory Processes: The Virtual Classroom and Office*. Proceedings of the International Conference on Disability, Virtual Reality and Associated Technology, Vesaprem, Hungary, September, 2002.

[Rizzo et al., 2006] Rizzo A.A; Bowerly T.; Buckwalter J.G.; Klimchuk D.; Mitura R.; Parsons T.D. *A virtual reality scenario for all seasons: the virtual classroom*. CNS Spectr. 11(1):35-44, 2006.

[RMI, 2007] Remote Method Invocation – RMI. http://java.sun.com/javase/technologies/core/basic/rmi/index.jsp. Último acesso em abril de 2007.

[Rosatelli e Tedesco, 2003] Rosatelli, M. C.; Tedesco, P. A. *Diagnosticando O Usuário Para Criação De Sistemas Personalizáveis*. In R. O. Anido & P. C. Masiero (Eds.), Anais do XXIII Congresso da SBC - III Jornada de Minicursos de Inteligência Artificial - MCIA, Vol. VIII, pp. 153-201. Porto Alegre: SBC

[Russell e Norvig, 2002] Russell, S. J.; Norvig, P. *Artificial Intelligence: A Modern Approach*. Norfolk: Prentice Hall, New Jersey, second edition, 2002. 1132p

[SAI, 2007] SAI Tutorial. http://www.xj3d.org/tutorials/general_sai.html. Último acesso em abril de 2007.

[Sales, 2006] Sales, M. V. G. S. *Uso de Agentes em Ambientes Virtuais Adaptativos – O Caso do VEPersonal* (Trabalho de Graduação). Centro de Informática – UFPE, Recife, 2006.

[Santos e Osório, 2004] Santos, C. T.; Osório, F. S. *An intelligent and adaptive virtual environment and its application in distance learning*. In: Proceedings of the Working Conference on Advanced Visual Interfaces. ACM Press, Gallipoli, Italy, pp. 362 – 365, 2004.

[Santos, 2004] Santos, C. T. *Um ambiente Virtual Inteligente e Adaptativo Baseado em Modelos de Usuário e Conteúdo* (Dissertação de Mestrado). São Leopoldo: PIPCA da UNISINOS, 2004.

[Sarmento, 2005] Sarmento, L. C. M. *Gerenciamento de Objetos de Realidade Virtual Reutilizáveis para Ambientes Virtuais de Ensino* (Trabalho de Graduação), CIn/UFPE, 2005. http://www.cin.ufpe.br/~tg/2005-1/lcms.pdf.

[Schwab e Kobsa, 2002] Schwab, I.; Kobsa, A. *Adaptivity through unobstrusive learning*. KI, Special Issue on Adaptivity and User Modeling 3, 5-9, 2002.

[SecondLife, 2007a] Second Life. http://secondlife.com. Último acesso em setembro de 2007.

[SecondLife, 2007b] What is Second Life? http://secondlife.com/whatis. Último acesso em setembro de 2007.

[Siedler, 2004] Siedler, M. S. *SIDATA – Sistema de Integração de Dados Apoiado no Tamino* (Dissertação de Mestrado) Recife: Pós-Graduação em Ciência da Computação do CIn/UFPE, 2004.

[SL, 2007] FIPA Communicative Act Library Specification. http://www.fipa.org/specs/fipa00037/SC00037J.html. Último acesso em setembro de 2007

[Solaris, 2007] Solaris 10 Operating System. http://www.sun.com/software/solaris. Último acesso em abril de 2007.

[SSL, 2007] Secure Sockets Layer. http://www.webopedia.com/TERM/S/SSL.html. Último acesso em abril de 2007.

[Struts2, 2007] Apache Struts 2. http://struts.apache.org/2.x/. Último acesso em setembro de 2007.

[Tamino, 2006] Technical Factsheet: Tamino XML Server. http://www.softwareag.com/Corporate/Images/FS_Tamino_ServerTech_100506_e_tcm16-5580.pdf. Último acesso em abril de 2007.

[Tamino, 2007] Tamino XML Server 4.2. http://www.softwareag.com/tamino. Último acesso em abril de 2007.

[Teichrieb, 1999] Teichrieb, V. *Avatares como Guias Interativos para Auxílio na Navegação em Ambientes Virtuais Tridimensionais* (Dissertação de Mestrado). Recife: Pós-Graduação em Ciência da Computação do CIn/UFPE, 1999.

[Teltzrow e Kobsa, 2004] Teltzrow M.; Kobsa A. *Impacts of user privacy preferences on personalized systems: a comparative study*. Designing personalized user experiences in eCommerce. Kluwer Academic Publishers, Norwell, MA, 2004.

[Tori e Kirner, 2006] Tori, R.; Kirner, C. *Fundamentos de Realidade Virtual*. In: Fundamentos e Tecnologia de Realidade Virtual e Aumentada. Eds. Romero Tori, Cláudio Kirner e Robson Siscoutto. Livro do Pré-Simpósio, VIII Symposium on Virtual and Augmented Reality, Belém – PA, 2006.

[Trevisan, 2004] Trevisan, D.G. Designing smooth connections between worlds. Conference on Human Factors in Computing Systems, CHI '04 extended abstracts on Human factors in computing systems. New York, NY, USA. ACM Press, pp. 1043 – 1044, 2004.

[Vinson, 1999] Vinson, N. *Design guidelines for landmarks to support navigation in virtual environments*. In Proceedings of CHI '99, ACM Press, pp. 278–285, 1999.

[Virvou e Chrysafiadi, 2006] Virvou, M.; Chrysafiad, K. A *Web-based Educational Application for Teaching of Programming: Student Modeling via Stereotypes*. Proceedings of the Sixth IEEE International Conference on Advanced Learning Technologies. Publisher IEEE Computer Society Washington, DC, USA, pp. 117-119, 2006.

[Virvou e Kabassi, 2004] Virvou, M.; Kabassi, K. *Adapting the human plausible reasoning theory to a graphical user interface*. IEEE Transactions on Systems, Man and Cybernetics, Part A, Volume: 34, Issue: 4. pp. 546-563, 2004.

[Vosinakis e Panayiotopoulos, 2005] Vosinakis, S.; Panayiotopoulos, T. *A Tool for Constructing 3D Environments with Virtual Agents*. Multimedia Tools and Applications, 25, pp. 253-279, 2005.

[VRML, 2007] VRML97 Functional specification. ISO/IEC 14772-1:1997. http://www.web3d.org/x3d/vrml. Último acesso em abril de 2007.

[W3C, 2007] Word Wide Web Consortion. http://www.w3c.org. Último acesso em abril de 2007

[Walczak e Cellary, 2002] Walczak, K.; Cellary, W. *Building database applications of virtual reality with x-vrml*. In Proceedings of the 7th International Conference on 3D Web Technology, pp. 111–120, 2002.

[Web3D, 2007a] Web3D Consortium. *Open Standards for Real-Time 3D Communication*. http://www.web3d.org. Último acesso em abril de 2007

[Web3D, 2007b] VRML Archives. http://www.web3d.org/x3d/vrml. Último acesso em setembro de 2007

[Webb et al., 2001] Webb, G.; Pazzani, M.; Billsus, D. *Machine Learning for user modeling*. User Modeling and User-Adapted Interaction, vol. 11, pp.19-29, 2001.

[Werner e Hanson, 1999] Wernert, E. A.; Hanson, A. J. *A framework for assisted exploration with collaboration*. In Proceedings of IEEE Visualization'99 (VIS '99), pp. 241–248. 1999.

[Windows, 2007] *Windows Products, Technologies, and Services*. http://www.microsoft.com/windows. Último acesso em abril de 2007.

[WindowsXP, 2007] *Windows XP*. http://www.microsoft.com/windows/products/windowsxp. Último acesso em abril de 2007.

[Wooldridge e Jennings, 1995] Wooldridge, M., Jennings, N. *Intelligent Agents: Theory and Practice*. In: Knowledge Engineering Review, vol. 10, no. 2, 1995.

[X3D, 2007a] X3D International Specification Standards. http://www.web3d.org/x3d. Último acesso em abril de 2007.

[X3D, 2007b] X3D Overview. http://www.web3d.org/x3d/specifications/x3d_specification.html. Último acesso em abril de 2007.

[X3D, 2007c] Overview of X3D Profiles and Conformance. http://www.web3d.org/about/overview. Último acesso em abril de 2007.

[Xj3D, 2007] Xj3D - Java based X3D Toolkit and X3D Browser. http://www.web3d.org/x3d/xj3d. Último acesso em abril de 2007.

[XML, 2007] Extensible Markup Language. http://www.w3.org/XML. Último acesso em abril de 2007.

[XMLSchema, 2007] XML Schema. http://www.w3.org/XML/Schema. Último acesso em abril de 2007.

[XP, 2007] Extreme Programming: A gentle introduction. http://www.extremeprogramming.org. Último acesso em setembro de 2007.

[XPath, 2007] XML Path Language – Xpath 2.0. http://www.w3.org/TR/xpath20. Último acesso em abril de 2007.

[XQL, 1999] XQL (XML Query Language). http://www.ibiblio.org/xql/xql-proposal.html. Último acesso em abril de 2007.

[XQuery, 2007] XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery. Último acesso em abril de 2007.

[XSD, 2007] XML Schema Definition. http://www.w3schools.com/schema/schema_intro.asp. Último acesso em abril de 2007.

[XSL, 2007] The Extensible Stylesheet Language Family – XSL. http://www.w3.org/Style/XSL. Último acesso em abril de 2007.

[XUpdate, 2005] XUpdate update, by Uche Ogbuji, 2005. http://www.oreillynet.com/onlamp/blog/2005/04/xupdate_update.html. Último acesso em abril de 2007.

[Zukerman e Albrecht, 2001] Zukerman, I.; Albrecht, D. W. *Predictive statistical models for user modeling*. User Modeling and User-Adapted Interaction Journal, Ed: Springer, 11(1-2), 5-18, 2001.

Apêndice 1 – Produção Científica

A produção científica desenvolvida durante o projeto de tese pode ser aferida com publicações de artigos em congressos e workshops, e também com o desenvolvimento de projetos de pesquisa, de iniciação científica e trabalhos de graduação realizados por alunos da Universidade Federal de Pernambuco e do UNIPÊ – Centro Universitário de João Pessoa – PB.

As publicações geradas durante os trabalhos desenvolvidos na elaboração desta tese são:

- AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C; SOUZA, D. A. C. M.; FUJIOKA, R. C.; VIEIRA, M. M. S. *An Infrastructure to Generate Adaptive Virtual Environments with the Management of Client-Server Communication in Real Time*. X Symposium on Virtual and Augmented Reality, SVR 2008, João Pessoa - PB, Brazil, May 13-16, 2008 (to appear).
- AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C; SOUZA, D. A. C. M.; FUJIOKA, R. C. *Supporting Adaptive Virtual Environments with Intelligent Agents.* 7th International Conference on Intelligent Systems Design and Applications, ISDA'07, Rio de Janeiro-RJ, Brazil, October, 22-24, 2007.
- AQUINO, M. S.; SALES, M. V. G.; SOUZA, F. F.; FRERY, A. C. *O Uso de Agentes Inteligentes na Personalização de Ambientes Virtuais.* In: II Workshop de Aplicações em Realidade Virtual, WARV 2006, Recife-PE. v. I. pp. 57-60, 2006.
- AQUINO, M. S.; NETO, D. R. M.; SOUZA, F. F.; FRERY, A. C. *Uma Interface 3D para ambientes Virtuais: o caso do VEPersonal*. In: II Workshop de Aplicações em Realidade Virtual, WARV 2006, Recife-PE. v. I. pp. 85-88, 2006.
- AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C.; NETO, L. G. A.; ALBUQUERQUE, M V. A.; ALMEIDA, R. M. G. *Adaptação de Conteúdos pelo Perfil do Usuário para Personalização de Ambientes Virtuais com X3D*. VIII Symposium on Virtual Reality, SVR 2006, Belém-PA, Brazil, May 02-05, 2006 (short paper).

- AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C. *VEPersonal – An infrastructure of Virtual Reality Components to Generate Web Adaptive Environments*. ACM International Conference Proceeding Series; Vol. 125. Proceedings of the 11th Brazilian Symposium on Multimedia and the Web, Poços de Caldas-MG, Brazil, December 05-07, pg. 1-8, 2005.
- AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C. *Three-dimensional Virtual Environments Adaptive to the Student's Profile for Distance Learning*. XVI Simpósio Brasileiro de Informática na Educação – SBIE 2005, Juiz de Fora - MG, Brasil, 09-11 Novembro, 2005.
- AQUINO, M. S.; SOUZA, F. F.; FRERY, A. C. *A Multi-Agent Architecture for Generating and Monitoring Adaptive Virtual Environments*. 5th International Conference on Hybrid Intelligent Systems – HIS'05, Rio de Janeiro-RJ, Brazil, November 06-09, p.515-517. Publisher: IEEE Computer Society, Washington, DC, USA, 2005 (short paper).

Trabalhos de iniciação científica e de conclusão de curso também foram realizados durante as pesquisas desenvolvidas nesta tese:

- *Integração da interface de comunicação com o servidor e validação dos módulos desenvolvidos no VEPersonal* (Projeto de Pesquisa). Daniel Abella C. M. de Souza, Rodrigo C. Fujioka e Marcelo Vieira. Curso de Computação da UNIPÊ, João Pessoa - PB, 2006.2 a 2007.2. Orientador Marcus S. Aquino.
- *Integrando Interface de Comunicação na Web em Ambientes Virtuais Adaptativos - Um Estudo no VEPersonal* (Trabalho de Graduação). Adson Luís Aguiar. Centro de Informática da UFPE, Recife - PE, 2006.2. Orientador Prof. Fernando F. Souza, co-orientador Marcus S. Aquino.
- *Uso de Agentes em Ambientes Virtuais Adaptativos – O Caso do VEPersonal* (Trabalho de Graduação). Marcus Vinicius Gonçalves Sales. Centro de Informática da UFPE, Recife- PE, 2006.1. Orientador Prof. Fernando da Fonseca de Souza, co-orientador Marcus S. Aquino.

- U*ma interface de comunicação para o ambiente VEPersonal utilizando Xj3D e a API Scene Authoring Interface* (Trabalho de Graduação). Domingos Rodrigues de Menezes Neto. Centro de Informática da UFPE, Recife - PE, 2006.1. Orientador Prof. Fernando da Fonseca de Souza, co-orientador Marcus S. Aquino.
- *Escolha entre Banco de Dados Relacional e Banco de Dados XML: Estudo de Caso para o Projeto VEPersonal* (Trabalho de Graduação). Maycon Vinícius A. De Albuquerque e Rodrigo Monteiro G. de Almeida. Curso de Computação da UNIPÊ, João Pessoa – PB, 2006.1. Orientadora Profa. Renata F. Viegas, co-orientador Marcus S. Aquino.
- *Ambientes Virtuais Adaptativos para Plataformas EAD: Criação de Ambientes Virtuais com Características Cognitivas* (Iniciação Científica). Emannuel Gomes Macedo. Centro de Informática da UFPE, Recife - PE, 2004.1. Orientador Prof. Fernando F. Souza, co-orientador Marcus S. Aquino.

Tese de Doutorado apresentada por Marcos Sidnei de Aguiar à Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título "*Virtuemul – Uma Infra-Estrutura para Criação e Manutenção de Ambientes Virtuais Adaptativos*", orientada pelo Prof. Alejandro César Frery Orgambide e aprovada pela Banca Examinadora formada pelos professores:

Profa. Judith Kelner
Centro de Informática / UFPE

Profa. Patrícia Cabral de Azevedo Restelli Tedesco
Centro de Informática / UFPE

Prof. Valmir Carneiro Barbosa
Centro de Informática / UFPE

Prof. [illegible]
[illegible]

Prof. [illegible]
[illegible]

Visto e permitida a impressão.
Recife, 3 de dezembro de 2007.

Prof. [illegible]
Coordenador da Pós-Graduação em Ciência da Computação
Centro de Informática da Universidade Federal de Pernambuco