VEGA – UM MODELO DE INTEGRAÇÃO DE APLICAÇÕES EXTERNAS A AMBIENTES VIRTUAIS ATRAVÉS DO USO DE APLICAÇÕES MEDIADORAS E DE SIMULÓIDES

por

Emanuele Marques dos Santos

Dissertação apresentada ao
Mestrado em Ciência da Computação
Universidade Federal do Ceará

Orientador: Creto Augusto Vidal, PhD

Dezembro, 2001

Agradecimentos

Agradeço, em primeiro lugar, a Deus por mais uma bênção derramada sobre a minha vida. À minha família, meu painho Raimundo, minha mainha Jaubeth e minha irmã Isabela, pelo carinho e apoio incansável. Ao professor Creto, pela confiança depositada em mim e no meu trabalho. Aos meus amigos, Júnior e Camilo, pela ajuda e pelos conselhos valiosos, sem a ajuda de vocês essa dissertação não seria possível. A todos os meus amigos e colegas que torceram por mim. À minha melhor amiga, Kel, pela compreensão, pelos maravilhosos momentos de descontração e, principalmente, por acreditar em mim. E, finalmente, a todos que contribuíram de alguma forma para que esse sonho se concretizasse. A todos o meu mais sincero obrigada!

Sumário

A realidade virtual tem sido explorada em um número cada vez maior de aplicações, permitindo a colaboração entre os participantes do ambiente virtual mesmo que eles se encontrem geograficamente distantes uns dos outros. Em paralelo ao desenvolvimento das aplicações de realidade virtual, houve um avanço acelerado de software especializado nas diversas áreas do conhecimento. Apesar da integração de aplicações externas a ambientes de realidade virtual mostrar-se como uma tarefa bastante complexa, ela pode ser resolvida através do uso de aplicações mediadoras. Portanto, essa dissertação propõe um modelo, denominado VEGA, para a integração de aplicações externas a ambientes virtuais baseado no conceito de aplicações mediadoras e no uso de personagens virtuais controlados por computador. Esse tipo de solução possibilita uma maior facilidade no processo de integração, pois se elimina a necessidade de reimplementação das aplicações originais. Para demonstrar a versatilidade do modelo proposto, foram implementadas duas aplicações: a primeira disponibiliza o serviço de dicionário eletrônico aos usuários do ambiente virtual e a segunda disponibiliza os serviços de um software de computação simbólica aos alunos de uma sala virtual de matemática. Utilizando-se do mesmo modelo de integração apresentado nesse trabalho, outros recursos importantes podem ser disponibilizados a usuários de sistemas de realidade virtual.

Summary

Virtual reality has been exploited in a large number of applications, which allow for the collaboration among participants even if they are remotely apart from one another. The development of specialized software in many areas has also experienced a considerable improvement in recent years. Although the integration of external applications with virtual environments seems to be a very complex task, it can be solved by the use of mediator systems. Therefore, this dissertation proposes a model, named VEGA, for the integration of external applications with virtual environments. VEGA is based on the concept of mediator systems and on the use of autonomous virtual characters. In order to demonstrate the versatility of the proposed model, two applications were implemented: the first application makes the services of electronic dictionaries available to the users of a virtual environment, while the second application enables the students and teachers of a virtual mathematics classroom to use a commercial symbolic computation software. This kind of solution makes the integration process easier because there is no need to rewrite the applications. By using the proposed model, other important resources can be made available to all of the users of virtual reality systems.

Índice

Lista de Anexos

Lista de Figuras

Lista de Tabelas

Capítulo I

Introdução

1.1 Motivação

Nos últimos anos, a realidade virtual tem sido explorada em um número cada vez maior de aplicações que possibilitam a colaboração entre os participantes dos ambientes virtuais [Kirner, 1998; Macedonia et al., 1995; Vidal, 1999]. Esses ambientes virtuais compartilhados permitem o desenvolvimento de uma gama de atividades, tais como discussões em uma sala de reuniões, estudos de protótipos, simulações militares, cirurgias virtuais colaborativas, atividades educacionais em laboratórios, explorações ecológicas para estudo de ecossistemas, entre outras atividades.

Em paralelo ao desenvolvimento das aplicações de realidade virtual cooperativas, houve um avanço acelerado de software especializado nas mais diversas áreas, tais como, a de banco de dados [Oracle Corporation], a de agentes inteligentes [Knapik & Johnson, 1998], a de cálculos computacionais e a de apoio às atividades de ensino [Maple, 2001; Matlab, 2001; Wolfram, 1996]. É importante observar que, quando se fizer necessário, não se pode deixar de aproveitar o grande potencial computacional dessas ferramentas nos sistemas de realidade virtual desenvolvidos para essas áreas de aplicação ou áreas correlatas.

No entanto, a integração de aplicações externas a ambientes virtuais experimenta o mesmo tipo de problema de integração de sistemas legados a sistemas modernos. Em algumas abordagens, para que essa integração ocorra de fato, são necessárias modificações quer no sistema de realidade virtual quer nas aplicações externas. Felizmente, a maioria dos sistemas de realidade virtual em rede fornece uma interface aberta para a comunicação com seus clientes, tais como o VLNET [Çapin et al., 1997] e o DIVE [Frécon & Stenius, 1998], que pode ser aproveitada para a integração com outras aplicações. O problema é que nem sempre as modificações nas aplicações externas para permitir sua utilização a partir dos ambientes virtuais são viáveis.

Assim como no caso de sistemas legados, a solução ideal para esse caso deveria basear-se no uso de mediadores. Muitas soluções baseadas no uso de mediadores têm sido desenvolvidas para a integração de sistemas e dados legados [Papakonstantinou et al., 1995; Souder & Mancoridis, 1999]. Soluções desse tipo permitem que as aplicações a serem integradas não precisem sofrer modificações, podendo até mesmo ser aplicações de plataformas diferentes.

1.2 Objetivos

No presente trabalho, propõe-se um modelo de integração de aplicações externas a ambientes virtuais através de aplicações mediadoras e personagens virtuais controlados por computador, o modelo VEGA. O VEGA é bastante portável e de fácil adaptação a diversos casos de integração. Por basear-se em mediadores, o modelo não exige que ocorram modificações nas aplicações que ele integra (ambiente virtual ou aplicações externas), permitindo o aproveitamento de sistemas legados.

1.3 Estrutura da Dissertação

A presente dissertação encontra-se organizada em seis capítulos e nove anexos, cujos conteúdos estão resumidos a seguir.

No Capítulo II, apresenta-se a tecnologia de realidade virtual, incluindo conceitos fundamentais e exemplos de aplicações baseadas nessa tecnologia. No Capítulo III, discutem-se as abordagens de integração de aplicações externas a ambientes virtuais. No Capítulo IV, apresenta-se o modelo VEGA de integração de aplicações externas a ambientes virtuais através de aplicações mediadoras e de simulóides. No Capítulo V, dois estudos de casos do modelo VEGA são amplamente detalhados. No Capítulo VI, apresentam-se algumas conclusões e enumeram-se alguns possíveis trabalhos futuros. O Anexo 1 contém o projeto de interfaces da aplicação BIA. O Anexo 2 ilustra o diagrama de classes da aplicação BIA, incluindo suas especificações. O Anexo 3 apresenta o diagrama de seqüência da aplicação BIA. O Anexo 4 contém o arquivo de script de ações do simulóide controlado pela aplicação BIA. O Anexo 5 apresenta o projeto de interfaces da aplicação Beremiz. O Anexo 6 ilustra o diagrama de classes da aplicação Beremiz, incluindo suas especificações. O Anexo 7 apresenta o diagrama de seqüência da aplicação Beremiz. O Anexo 8 contém o arquivo de script de ações do simulóide controlado pela aplicação Beremiz. E, finalmente, o Anexo 9 apresenta os arquivos com os dados dos mapeamentos das funções do Mathematica.

Capítulo II

Realidade Virtual

Nesse capítulo são discutidos os termos e as idéias básicas que envolvem a tecnologia de Realidade Virtual (RV). Inicialmente, são explicados os conceitos de realidade virtual, mostrada uma visão geral dos tipos de ambientes virtuais e seus componentes e, finalmente, são discutidos exemplos de aplicações de realidade virtual em algumas áreas do conhecimento.

2.1 O que é realidade virtual?

Há várias definições aceitas para realidade virtual. [Hancock, 1995] define realidade virtual como a forma mais avançada de interface do usuário de computador até agora disponível. Por sua vez, [Leite Jr, 2000], a partir de [Burdea et al., 1994; Jacobson, 1991; Kirner, 2001; Krueger, 1991] entende realidade virtual como uma forma avançada de interface, onde o usuário pode realizar imersão, navegação e interação em um ambiente sintético tridimensional gerado por computador, utilizando canais multi-sensoriais em tempo real.

De acordo com o grau de imersão fornecido, os sistemas de realidade virtual podem ser classificados como imersivos e não-imersivos.

Os sistemas imersivos utilizam dispositivos não-convencionais de entrada e saída, tais como: capacete de visualização ou sala com projeções das visões nas paredes, teto, e piso [Cruz-Neira et al., 1992]; luva-de-dados (*dataglove*); dispositivos de som [Begault, 1994]; sensores que rastreiam os movimentos do corpo; etc. Exemplos de alguns desses dispositivos podem ser vistos nas figuras 1 e 2.

Os sistemas de realidade virtual não-imersivos fazem uso de dispositivos comuns de entrada e saída, tais como monitor, mouse e teclado. Embora a realidade virtual imersiva represente o tipo ideal de realidade virtual, principalmente devido ao melhor sentimento de imersão que ela proporciona, a realidade virtual não-imersiva apresenta grandes vantagens graças ao seu baixo custo e à sua facilidade de uso. Além disso, dispositivos baseados em outros sentidos, principalmente o auditivo (dispositivos de som),

acabam contribuindo para a obtenção de um grau de imersão aceitável. Essas características da realidade virtual não-imersiva, também conhecida como *desktop virtual reality*, foram responsáveis pela popularização de seu uso.

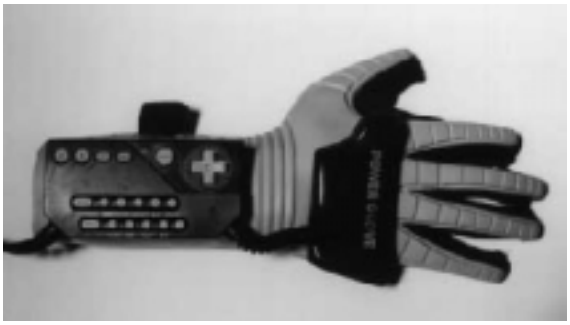


Figura 1. Uma PowerGlove: luva-de-dados da *Nintendo Home Entertainment System*.

Figura 2. Uso de capacete de visualização e luvas.

O ambiente sintético tridimensional gerado por computador, que possibilita o usuário visualizar, interagir e manipular dados, é chamado de ambiente virtual ou mundo virtual.

2.2 Ambientes Virtuais

Os ambientes virtuais propiciados por um sistema de realidade virtual podem simular tanto um espaço do mundo real, quanto o espaço de um mundo imaginário, no qual o usuário possa sentir-se inserido [Machado, 1995]. Os ambientes virtuais consistem em espaços tridimensionais e suas entidades componentes.

2.2.1 Entidades presentes em ambientes virtuais

Existem três tipos de entidades presentes nos ambientes virtuais: os avatares, os simulóides e os objetos.

Avatares

Avatares são representações gráficas tridimensionais dos usuários. Além de representá-lo, o avatar também é o meio pelo qual o usuário pode estabelecer relações interativas com todo o mundo virtual, sentindo-se mais presente dentro desse ambiente [Slater & Usoh, 1994]. Os avatares podem apresentar qualquer corpo tridimensional animado dependendo do que se propõe o ambiente virtual em questão. Um exemplo bem interessante da utilização de avatares é a aplicação de bate-papo baseada em realidade virtual chamada Papo3D (http://www.papo3d.com.br) (Figura 3).

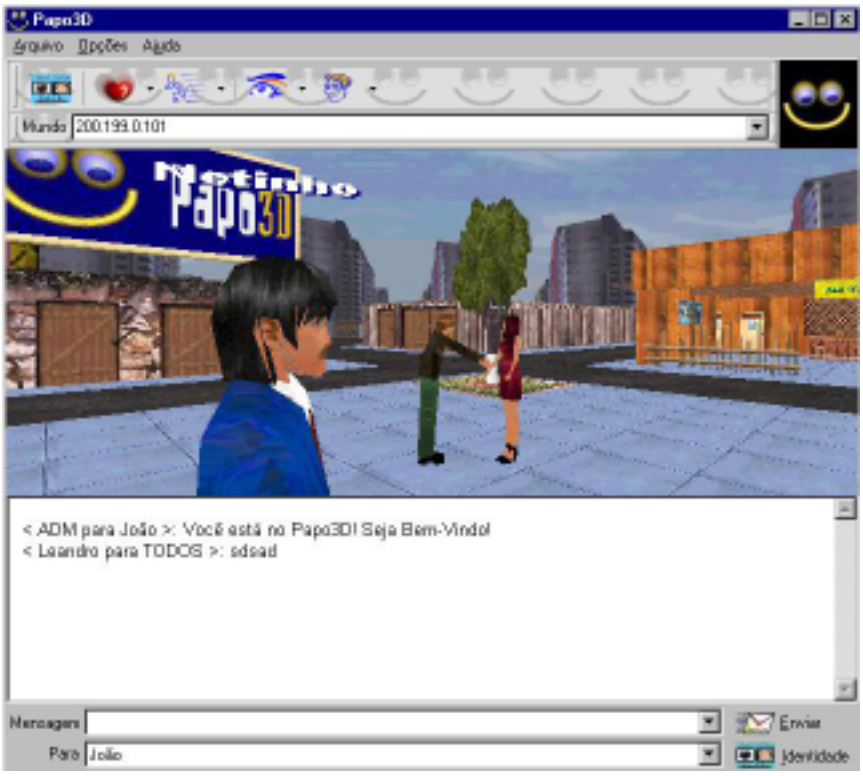


Figura 3. Janela da aplicação Papo3D.

Nessa aplicação, cada usuário escolhe um avatar dentre os vários disponíveis para representá-lo dentro do ambiente virtual e, uma vez conectado ao ambiente, pode alterar a aparência física de seu avatar como, por exemplo, tornando-o mais magro ou mais gordo, mais alto ou mais baixo. A navegação pelo ambiente utilizando-se um avatar pode dar-se em primeira pessoa (o usuário vê o ambiente sob a perspectiva do próprio avatar) ou em

terceira pessoa (o usuário vê o ambiente do ponto de vista de um observador que segue seu avatar). Além disso, o avatar pode realizar outras ações, tais como acenar, cumprimentar, beijar e dançar. Na Figura 3, o usuário João, navegando em primeira pessoa, observa o avatar de “Netinho” à sua frente, enquanto os avatares de “Gil” e “Ana” cumprimentam-se.

Simulóides

Simulóides são personagens virtuais controlados por computador [Csordas, 1997]. Assim como os avatares, os simulóides podem apresentar qualquer corpo tridimensional. Ademais, podem interagir tanto com avatares, quanto com outros simulóides, e são capazes de executar desde tarefas simples e repetitivas pré-programadas até tarefas complexas que requerem um certo grau de inteligência na sua realização. Um exemplo do uso de simulóides pode ser observado no projeto Campus Virtual da UFPE (http://www.cin.ufpe.br/~if291/galeria/ufpe/cufpe.wrl), desenvolvido no Centro de Informática da UFPE. Esse projeto consiste de um mundo virtual tridimensional que representa o campus universitário e possui dois simulóides, uma abelha e uma formiga (Figura 4). Esses simulóides têm como objetivo mostrar o campus para o visitante, auxiliando-o na navegação e obtenção de informações.

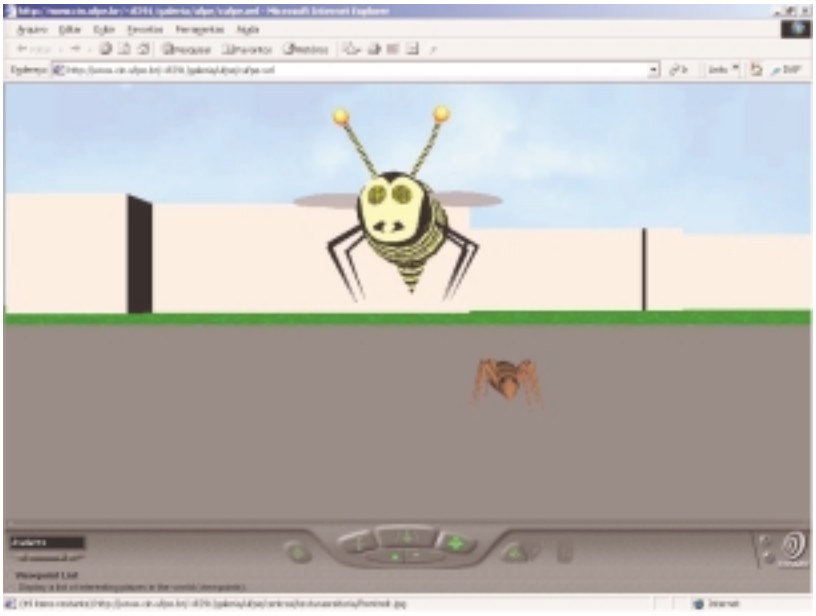


Figura 4. Simulóides guias do Campus Virtual da UFPE.

Objetos

Alguns dos objetos encontrados dentro de ambientes virtuais são simples corpos estáticos, como é o caso de pias e balcões, por exemplo. Outros objetos apresentam

elaborados comportamentos dinâmicos, como os instrumentos de precisão encontrados em laboratórios virtuais. Segundo a capacidade de interação dos objetos (definida de acordo com o papel desempenhado dentro do ambiente virtual), eles podem ser classificados como não-interativos e interativos.

Os objetos não-interativos são concebidos para não reagirem a ações de outras entidades do ambiente virtual. Por exemplo, peças de decoração, sinais de indicação em determinados locais podem ser considerados objetos não-interativos em certas aplicações.

Por outro lado, objetos interativos reagem a eventos causados por outras entidades, permitindo em alguns casos, serem manipulados. Um exemplo de objeto interativo é um béquer em um laboratório virtual de química. O comportamento desse objeto precisa ser bem elaborado: incorporar um mecanismo de detecção de aproximação de objetos, controlar o conteúdo no seu interior e, principalmente, verificar a ocorrência, ou não, de reações químicas. Outros exemplos de objetos interativos são o quadro branco virtual, que possibilita o compartilhamento de uma determinada área gráfica na qual indivíduos inserem e manipulam, de forma cooperativa, imagens, objetos gráficos e textos; e o projetor virtual de slides, através do qual é possível criar e exibir apresentações contendo textos tridimensionais (Figura 5).

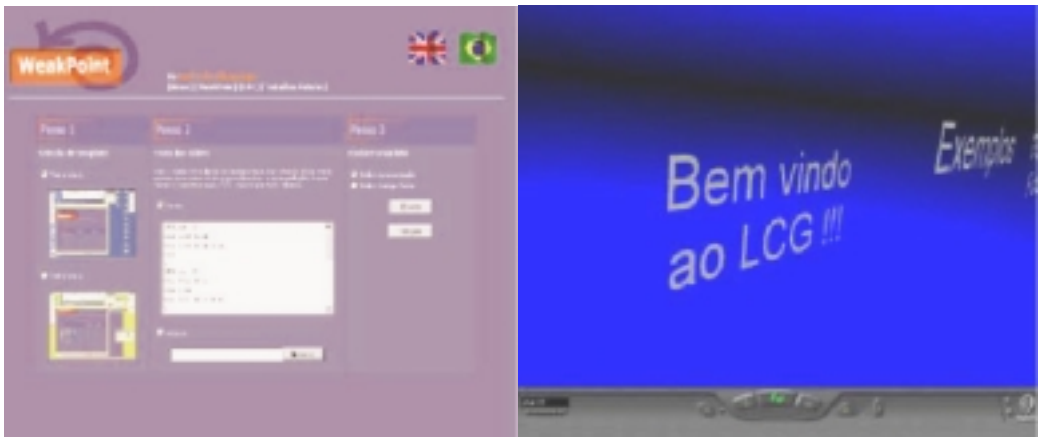


Figura 5. Ferramenta para edição e visualização de apresentações de textos tridimensionais.

2.2.2 Tipos de ambientes virtuais

De um modo geral, as aplicações de realidade virtual podem variar bastante, desde aplicações onde uma única pessoa usa apenas um computador, até aplicações em que muitos usuários interagem em um sistema distribuído. É possível classificar os ambientes virtuais em: monousuários e multiusuários.

Um ambiente virtual monousuário, como o próprio nome já o diz, permite sua utilização por apenas um usuário por vez. Ambientes desse tipo são amplamente utilizados para visualização de dados e são facilmente encontrados na *web*, podendo ser acessados por intermédio de um navegador especial, geralmente VRML (*Virtual Reality Modeling Language*), que já vem acoplado aos principais navegadores HTML.

Um ambiente multiusuário, por sua vez, permite um ou mais usuários ao mesmo tempo dentro de sua estrutura. Esses ambientes são mais interessantes porque possibilitam a interação entre os usuários. Dependendo do ambiente, essa interação pode se dar tanto através da comunicação entre usuários; quanto através do compartilhamento do próprio ambiente virtual; ou da realização de trabalho cooperativo.

Os ambientes virtuais multiusuários mais importantes são os ambientes virtuais em rede (AVRs), pois permitem a colaboração entre participantes mesmo quando encontram-se geograficamente dispersos. Tais ambientes são descritos a seguir.

2.3 Ambientes Virtuais em Rede (AVRs)

AVRs são sistemas que permitem a interação entre vários usuários em tempo real dentro de um ambiente virtual compartilhado mesmo que esses usuários estejam geograficamente distantes uns dos outros [Singhal & Zyda, 1999].

Os AVRs mais modernos e eficientes são distribuídos. Eles são considerados um dos sistemas de software mais complexos já construídos [Stytz, 1996]. Nesses sistemas, cada participante faz uso de uma cópia específica do ambiente virtual. Sempre que as interações dentro desse ambiente provocarem alterações no estado de uma das cópias do ambiente virtual compartilhado, informações são enviadas automaticamente aos computadores utilizados pelos participantes, para que as cópias do ambiente virtual sejam atualizadas. Dessa forma, a consistência geral do AVR é mantida e os participantes têm a impressão de compartilhar um único ambiente virtual. Essa impressão de compartilhamento estimula fortemente a colaboração entre os participantes para a execução das mais variadas tarefas [Çapin et al., 1999].

Um ambiente virtual em rede possui cinco características que o diferenciam dos demais.

- O sentimento compartilhado de espaço: todos os participantes possuem a ilusão de estarem situados no mesmo lugar, tal como na mesma sala, edifício ou terreno. Esse espaço compartilhado representa um local comum (real ou fictício) dentro do qual outras interações podem acontecer. O espaço

compartilhado deve apresentar as mesmas características a todos os participantes. Por exemplo, todos os participantes devem perceber as mesmas temperaturas e condições climáticas, assim como a mesma acústica [Singhal & Zyda, 1999]. Apesar de não serem necessariamente apresentados graficamente, os AVRs mais eficientes provêem uma representação gráfica tridimensional do espaço compartilhado.

- O sentimento compartilhado de presença: ao entrar no espaço compartilhado, cada participante é representado por um avatar e pode ver os outros avatares localizados no mesmo espaço compartilhado, da mesma forma que os outros usuários vêem o avatar do novo participante. Analogamente, quando o usuário sai do espaço compartilhado, os outros avatares devem ver o avatar sumir.
- O sentimento compartilhado de tempo: participantes devem estar aptos a ver o comportamento uns dos outros ao mesmo tempo em que ele ocorre, ou seja, o AVR deve possibilitar que aconteça interação em tempo real.
- A existência de um meio de comunicação: apesar da visualização ser a base de um AVR eficiente, a maioria dos AVRs também possibilita a comunicação entre os participantes, através de gestos, texto ou voz. Essa comunicação adiciona um sentimento necessário de realismo a qualquer ambiente simulado, sendo um componente fundamental de sistemas de engenharia ou de treinamento.
- A possibilidade de compartilhamento: o verdadeiro poder dos AVRs vem da habilidade dos usuários interagirem realisticamente tanto entre si quanto com o próprio ambiente virtual. Na simulação de uma batalha ou de um jogo, por exemplo, os usuários precisam atirar uns nos outros, ou podem colidir uns com os outros. Os usuários devem ser capazes de manipular objetos que existem no ambiente, assim como repassá-los a outros participantes. Um projetista de AVRs deveria até mesmo possibilitar aos usuários manipular o próprio ambiente. Por exemplo, construir um abrigo, desenhar em um quadro branco, ou até mesmo destruir o ambiente.

2.4 Aplicações de Realidade Virtual

Embora tenha sido empregada inicialmente no treinamento militar e nos jogos em rede [Singhal & Zyda, 1999], a realidade virtual tem sido utilizada em uma série de

aplicações [Mine, 1997], tais como:

- treinamento militar [Macedônia et al., 1995];
- terapia de fobias [Rothbaum et al., 1995];
- entretenimento [Pausch et al., 1996]; e
- escoamento de fluidos [Bryson & Levit, 1992].

Além dessas aplicações, alguns autores [Doenges et al., 1997; Zyda & Sheehan, 1997] afirmam que AVRs têm sido empregados com sucesso em um grande número de aplicações colaborativas, como por exemplo:

- teleconferência virtual com troca de objetos multimídia;
- trabalhos colaborativos envolvendo modelos tridimensionais;
- ambientes de jogos multiusuários;
- *teleshopping* envolvendo modelos tridimensionais, imagens, sons;
- aplicações médicas (diagnósticos a distância, cirurgias virtuais para treinamento);
- aprendizagem e treinamento a distância;
- agência de viagens virtual;
- estúdio virtual de vídeo com integração de mídias através de rede.

Dentre essa gama de aplicações que utilizam realidade virtual, algumas merecem ser vistas com mais detalhes devido ao seu caráter multidisciplinar. Elas estão divididas em dois grupos: os museus virtuais e os simulóides de conversação.

2.4.1 Museus Virtuais

De acordo com [Teichrieb, 1999], um bom museu virtual deveria possibilitar aos usuários fazerem tudo o que eles seriam capazes de fazer caso estivessem visitando um bom museu do mundo real, ou seja, visitar uma exposição interessante, parar na frente do quadro do qual se gosta, chegar mais perto, ler a legenda; percorrer o acervo à toa ou ir atrás de alguma curiosidade especial; passear pelo jardim, comprar um cartão postal e partir sentindo-se mais informado e feliz. Tudo isso com as vantagens dele não cobrar ingresso (possivelmente), estar sempre aberto, não ter filas, e estar a menos de um metro de onde se está sentado.

Muitos museus virtuais, tridimensionais ou não, têm sido construídos na *web* nos últimos anos, porém nem sempre se apresentam de forma tão intuitiva quanto se espera.

São descritas a seguir três aplicações de museu virtual: o Tour Virtual do Louvre, o Museu Virtual Guggenheim de Bilbao e o Projeto Museu Virtual.

Tour Virtual do Louvre

A página *web* de um dos museus mais famosos do mundo possibilita ao usuário que a acessa, visitar virtualmente nove de suas exposições (http://www.louvre.fr). Para chegar à página com as opções de galerias a serem visitadas, é só pressionar *Virtual Tour* no menu de navegação da página principal. Para poder visualizar as exposições é necessário que o usuário tenha instalado em seu computador o *QuickTime Player*, da *Apple* (http://www.quicktime.com).

O ambiente virtual de cada exposição (Figura 6) consiste em um vídeo projetado na superfície interna de um cilindro. O usuário, posicionado no interior do cilindro, terá uma visão de 360 graus da galeria. Porém, não há nenhuma possibilidade de interação entre o usuário e o ambiente. Os controles se limitam a girar para a direita ou para a esquerda; olhar para cima ou para baixo ou efeitos de *zoom.* Essa restrição de liberdade pode até contribuir para um sentimento de frustração por parte do usuário, principalmente acrescentado ao fato de que não se pode nem mesmo ler a legenda de algumas obras importantes. Isso impede que pessoas que não têm condições de ir ao Louvre, especialmente crianças de escolas com acesso à internet da América Latina, possam enriquecer suas visitas virtuais obtendo informações mais detalhadas sobre as obras do acervo.

Figura 6. Visita à primeira galeria de esculturas de Michelangelo.

Museu Virtual Guggenheim de Bilbao

O Museu Virtual Guggenheim de Bilbao é uma aplicação desenvolvida durante o projeto CoNViRA (*Conceptual Navigation in Virtual Reality Applications*) [Frery et al., 1999] de cooperação entre o Centro de Informática da UFPE e o *Research Institute for Computer Architecture and Software Technology* do *German National Research Center for Information Technology*.

O Museu Guggenheim de Bilbao tem uma estrutura grande e complexa, o que dificulta sobremaneira a orientação e a locomoção de seus visitantes. Visando amenizar esses problemas, [Teichrieb, 1999] apresenta uma metodologia para auxiliar a navegação em ambientes virtuais tridimensionais. A alternativa adotada, utilizando simulóides inteligentes, corresponde a *tours* guiados disponíveis aos visitantes das exposições do museu.

Essa aplicação consiste em uma versão simplificada do museu cujo conteúdo é fictício em função das restrições relacionadas ao fornecimento de informações sobre os objetos expostos no museu em Bilbao. Além de uma exposição que mostra todos os modelos tridimensionais dos museus Guggenheim existentes na atualidade no mundo (exceto o Guggenheim de Bilbao), o museu contém outras exposições fictícias baseadas nas obras expostas no Museu Virtual do Homem do Nordeste [Frery & Kelner, 2001], desenvolvido pelo Centro de Informática da UFPE, um museu antropológico real situado na cidade de Recife, em Pernambuco. Os usuários podem acessar informações suplementares através dos vários ícones tridimensionais espalhados pelo museu. Tanto as exposições quanto os ícones tridimensionais foram modelados utilizando a linguagem VRML e as informações suplementares constituem documentos bidimensionais construídos utilizando HTML e *Macromedia Flash*.

Habitando o museu virtual, existem quatro simulóides com a forma de peixes. Esses simulóides possuem características físicas e comportamentos específicos que lhes dão realismo e são considerados inteligentes, já que possuem conhecimento tanto sobre o ambiente em que vivem, como também sobre o usuário que está solicitando seu auxílio. Esses componentes devem ser embutidos no simulóide segundo o domínio da aplicação na qual serão utilizados.

A primeira etapa pela qual o usuário passa quando acessa a aplicação é a de preenchimento dos campos da interface a fim de coletar informações para a modelagem do seu perfil (Figura 7).

Figura 7. Janela da interface da aplicação Museu Virtual Guggenheim.

Esse perfil do usuário tem por objetivo definir o tipo de informação que ele espera explorar no ambiente e se ele deseja ser guiado por um simulóide. Se o usuário desejar ser guiado, ele deve escolher um dos quatro guias disponíveis no ambiente virtual (Figura 8).

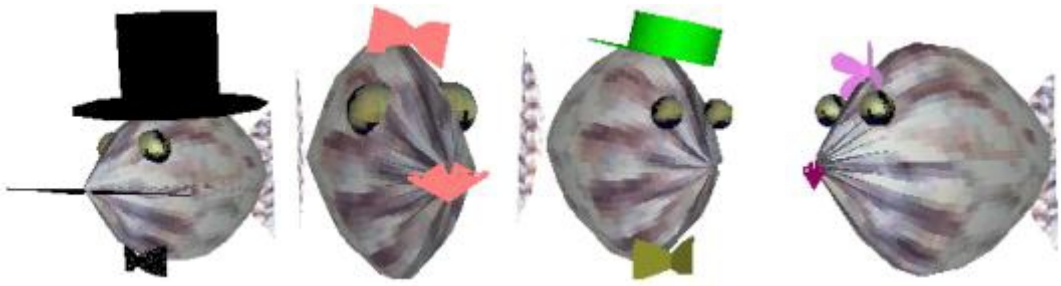


Figura 8. Simulóides-guia, da esquerda para a direita: pai, mãe, filho e filha.

A forma de visualização do conteúdo da exposição durante o tour guiado está associada ao peixe-guia selecionado. Por exemplo, o peixe mais novo pode mostrar uma exposição ao usuário sob a perspectiva de uma criança, visualizando o conteúdo a partir de uma posição mais baixa, correspondendo à altura de uma criança. Cada peixe-guia fará uso de rotas de navegação predefinidas.

Projeto Museu Virtual

Dentre os objetivos do Projeto Museu Virtual [Wazlawick, 1999], está incluído o desenvolvimento de uma ferramenta de autoria, que possibilite a criação de museus virtuais por estudantes, de forma colaborativa e de acordo com os princípios do construtivismo. Os museus virtuais criados por essa ferramenta de autoria podem ser organizados em salas com diferentes exposições onde os objetos tridimensionais em exibição estão disponíveis para serem diretamente explorados.

Essa ferramenta de autoria é colaborativa e possibilita a criação de museus, nos quais os usuários podem caminhar livremente dentro do espaço tridimensional e explorá-lo segundo as características de navegação, imersão e interação propiciadas pela realidade virtual. Cada usuário (estudante ou visitante) pode escolher o seu avatar, passear pelo mundo virtual, observar as atividades e ações realizadas e participar da criação ou modificação do museu virtual. Os usuários podem interagir entre si ou com guias virtuais através de texto e também podem visualizar informações relevantes para o contexto do museu apresentadas em forma tridimensional.

A ferramenta de autoria também permite a criação de simulóides ou guias virtuais, os quais possuem conhecimento sobre os objetos que são exibidos e sobre os visitantes do museu. Os simulóides guiam os usuários nas suas interações com objetos e avatares. Esses guias estão aptos a responder perguntas, apresentar sugestões para os visitantes do museu, realizar ações tais como caminhar de uma sala para outra e mover objetos. Eles também aprendem com os visitantes, isto é, descobrem seus nomes, seus interesses, suas localizações correntes no mundo virtual, etc.

2.4.2 Simulóides de Conversação

Os simulóides de conversação são meios poderosos de se estabelecer a interação entre os humanos e os computadores. Eles não são meras interfaces representadas por corpos de humanos ou de animais, mas também comportam-se da mesma forma que os humanos quando conversam entre si, principalmente a forma como eles usam o corpo na conversação.

Alguns simulóides desse tipo são também utilizados em ambientes voltados para treinamento. Nesse caso, os simulóides possuem o papel de auxiliar o estudante quando necessário.

A seguir são descritos três exemplos de simulóides. O primeiro, Steve, é designado para auxiliar no treinamento de tripulantes de embarcações. O segundo, Rea, é uma agente

imobiliária encarregada de sugerir imóveis de acordo com as necessidades do usuário. O terceiro trata-se de Herman, o inseto. Herman tem a função de ajudar estudantes com conceitos de anatomia e fisiologia botânica.

Steve (*Soar Training Expert for Virtual Environments*)

Steve (Figura 9) é o nome de um simulóide que ensina estudantes a desempenharem tarefas físicas e baseadas em instruções, tais como operar uma máquina complexa e realizar serviços de reparos e manutenção [Johnson et al., 1998; Johnson & Rickel, 1998; Rickel & Johnson, 1997, 1999].

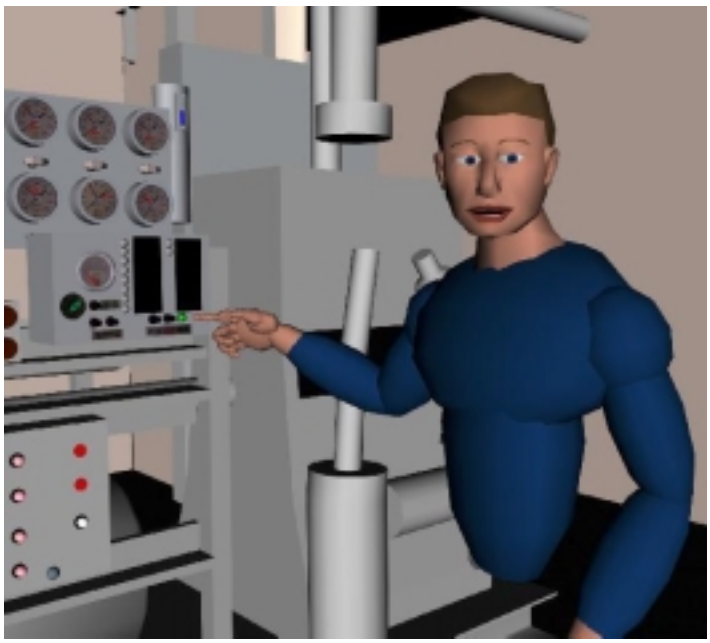


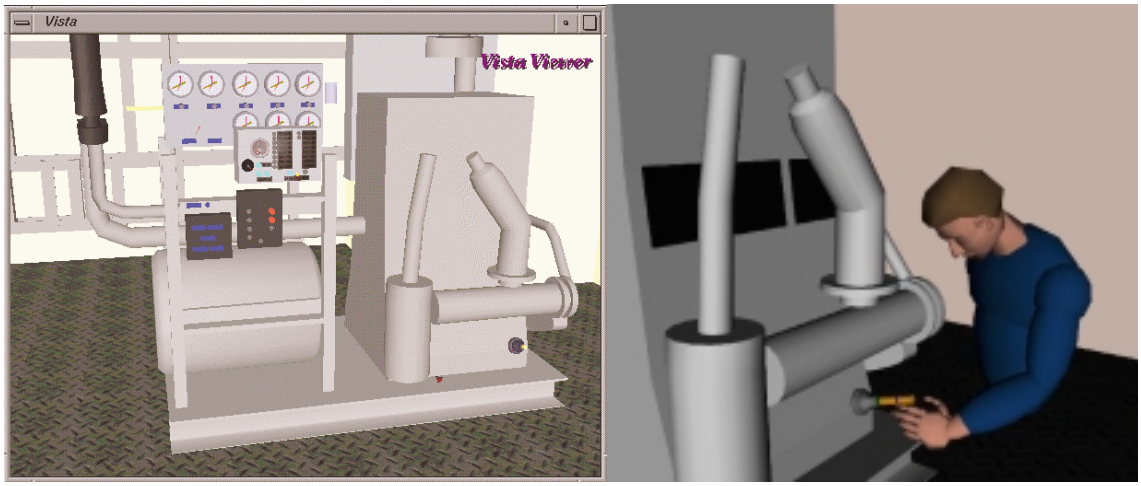
Figura 9. Steve mostrando ao estudante onde está a luz de força.

Steve e o estudante coabitam um ambiente tridimensional que simula o ambiente real de trabalho do estudante. Steve pode tanto demonstrar como realizar tarefas, como também monitorar estudantes enquanto eles as praticam, fornecendo assistência quando necessário. Além disso, Steve é projetado para coexistir com outros avatares e simulóides num mundo virtual. O objetivo dessa aplicação é dar suporte a treinamento em equipe, onde equipes de pessoas, possivelmente em locais diferentes, podem habitar o mesmo mundo virtual e aprender a realizarem tarefas como uma equipe. Portanto, Steve pode agir tanto como um tutor para membros individuais, como tomar o lugar de membros da equipe que estão ausentes.

Esse simulóide possui as capacidades pedagógicas esperadas de um sistema inteligente de tutoria. Por exemplo, ele pode responder perguntas, tais como “O que devo

fazer a seguir?” e “Por quê?”. Ademais, ele pode demonstrar ações, usar o olhar e gestos para chamar a atenção do estudante, gerar e reconhecer falas, adaptar procedimentos do domínio a eventos não esperados, lembrar de ações passadas, além de guiar o estudante pelo mundo virtual.

A primeira aplicação de Steve foi ensinar estudantes a operar o sistema da turbina propulsora de combustível a bordo das embarcações da marinha americana. A Figura 10b mostra Steve trabalhando com um Compressor de Ar de Alta Pressão (CAAP), que é parte do sistema de propulsão. Usando um capacete de visualização, os estudantes entram no lugar contendo o CAAP. A empresa *Lockheed-Martin* criou tanto o modelo tridimensional do CAAP (Figura 10a) e de sua vizinhança, quanto a aplicação de software *Vista Viewer* [Stiles et al., 1995], através da qual os estudantes visualizam e interagem com o mundo virtual. Os estudantes interagem com o equipamento tocando os diversos controles (por exemplo, válvulas e botões) usando uma luva de realidade virtual.



(a) (b)

Figura 10. (a) Modelo 3D do Compressor de Ar de Alta Pressão. (b) Steve mostrando como verificar o nível de óleo.

O conhecimento do Steve é formado a partir de um determinado domínio. Para ensinar estudantes tarefas pertencentes a um novo domínio, alguém deve fornecer o conhecimento desse domínio em uma linguagem declarativa. Os autores do Steve assumem que esse alguém será uma pessoa com experiência em criação de cursos naquele domínio, mas sem a necessidade de que ela tenha conhecimento de programação. A maior limitação do projeto do Steve é o fato de que ele está totalmente dependente dos tipos de conhecimento que um autor de curso pode fornecer.

Rea (*Real Estate Agent*)

Rea é um simulóide especialista na área imobiliária. Ela tem acesso a um banco de dados de apartamentos e casas disponíveis para venda em Boston. Ela pode mostrar imagens dessas propriedades e dos seus vários cômodos, apontar e discutir as características principais desses imóveis [Cassell, 2000a].

Como pode ser visto na Figura 11, Rea possui a forma humanóide e usa o seu corpo de uma maneira tipicamente humana durante a conversação, isto é, ela usa olhar, postura corporal, gestos e expressões faciais para, além de enriquecer os diálogos, organizar e regular a conversação. Ela também entende esses mesmos recursos quando empregados pelo seu interlocutor humano.

Figura 11. Rea exibindo a sala de estar de um apartamento.

Para que a Rea demonstrasse esses padrões de conversação bem pertinentes ao ser humano, foi necessário desenvolver um modelo conversacional que se baseia na função tanto dos comportamentos não-verbais como dos comportamentos verbais, e que torna explícitas as contribuições interativas e proposicionais desses comportamentos conversacionais [Cassel, 2000b].

A implementação atual dessa aplicação consiste em uma grande tela de projeção na qual Rea é exibida. O usuário se posiciona na frente dessa tela (Figura 12). Duas câmeras posicionadas no topo da tela de projeção rastreiam as posições da cabeça e das mãos do usuário no espaço. Os usuários usam um microfone para a captura da fala. Um único

computador *SGI Octane* executa o processamento gráfico e o mecanismo de conversação, enquanto vários outros computadores gerenciam o reconhecimento e a geração de falas e o processamento de imagens.

Figura 12. Rea interagindo com um usuário.

Herman, o inseto

Herman é um simulóide pedagógico (Figura 13) que habita um ambiente virtual para o aprendizado construtivista do domínio de anatomia e fisiologia botânicas para estudantes de primeiro grau. Ele faz parte de um projeto multidisciplinar de larga escala chamado *Design-A-Plant* [Lester & Stone, 1997; Lester et al., 1996, 1997a, 1997b, 1997c, 1999; Stone & Lester, 1996] e, interativamente, oferece aconselhamento contextualizado para os estudantes enquanto eles montam plantas graficamente a partir de uma biblioteca de estruturas de plantas, tais como raízes e caules.

O simulóide é um inseto falante e atraente com propensão a voar pela tela e mergulhar nas estruturas da planta enquanto dá conselhos para a solução do problema. Enquanto Herman explica algum dos conceitos, ele não se mantém imóvel, podendo caminhar, voar, encolher-se, esticar-se, nadar, pescar, teletransportar-se, fazer acrobacias e até pular de *bungee jumping*. No total, ele possui mais de 50 comportamentos animados e aproximadamente 100 comportamentos verbais.

Esse ambiente é composto por episódios nos quais os estudantes devem solucionar problemas. O objetivo do estudante a cada episódio é formar uma planta que se

desenvolverá em um dado ambiente com condições específicas (visualizado como um planeta imaginário diferente), tal como a quantidade de luz solar disponível. À medida que os estudantes resolvem os problemas construindo plantas, o simulóide oferece-lhes conselhos sobre anatomia e fisiologia botânicas.

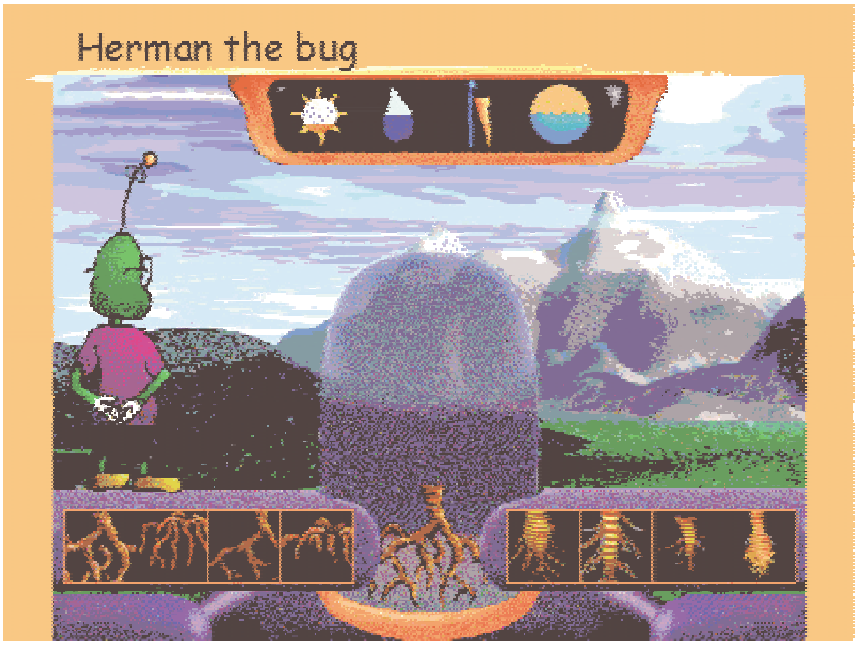


Figura 13. Herman, o inseto observando o usuário agindo.

Design a Plant possui os seguintes elementos:

- ambientes – 16 ambientes (4 tipos de ambientes, cada qual com 4 níveis de complexidade);
- biblioteca de componentes – 8 tipos de raízes, 8 tipos de caules e 8 tipos de folhas;
- modelo do domínio – 31 pacotes de restrições que relacionam 6 fatores ambientais às estruturas anatômicas e funções fisiológicas.

No início de cada episódio, os estudantes não estão familiarizados com o problema que vão resolver, portanto o simulóide assume o controle da situação e introduz o problema descrevendo as condições ambientais daquele ambiente específico sob as quais a planta será construída. O simulóide monitora os estudantes, enquanto eles constroem as plantas, e intervém, fornecendo explicações sobre anatomia e fisiologia botânicas, quando eles chegam a um impasse. Nesse caso, o simulóide está preparado para auxiliar os estudantes mesmo que não tenha havido um pedido de ajuda explícito. No final de cada

episódio, cujo problema foi resolvido, Herman parabeniza os estudantes e anda de um lado para o outro da tela fazendo cambalhotas.

2.4.3 Considerações sobre as aplicações de realidade virtual

Tanto os museus virtuais quanto a utilização de simulóides de conversação constituem aplicações de realidade virtual de grande importância devido ao seu caráter multidisciplinar. Além disso, observou-se através dos exemplos de aplicações citados acima que a possibilidade de disponibilizar informações nessas aplicações é muito grande, principalmente se os usuários pudessem, através dessas aplicações, utilizar os serviços disponíveis por outras ferramentas que não estão presentes dentro do ambiente virtual, tais como ferramentas de simulação, enciclopédias, dicionários e outras.

Capítulo III

Abordagens de Integração de Aplicações Externas a Ambientes Virtuais

A integração de aplicações externas a ambientes virtuais pode ser classificada em quatro abordagens de acordo com a ênfase da integração. Dessa forma, são apresentadas nesse capítulo as características, vantagens e desvantagens de cada uma das abordagens de integração de aplicações externas a ambientes virtuais.

3.1 Integração monolítica

Essa integração é denominada de monolítica porque o que existe de fato é uma absorção da aplicação externa pelo ambiente de realidade virtual. Nesse caso, o sistema de realidade virtual é único e está localizado em um computador que possui apenas um processador. Esse sistema gerencia todas as tarefas e os recursos necessários para sua execução, inclusive os serviços oferecidos pelas aplicações externas. Assim, os serviços precisam estar inseridos dentro do ambiente virtual de forma que o próprio sistema de realidade virtual controle o acesso a eles (Figura 14). É importante salientar que uma condição fundamental para esse tipo de integração é que tanto o sistema de realidade virtual quanto a aplicação externa sejam escritos nas mesmas plataformas de hardware e software.

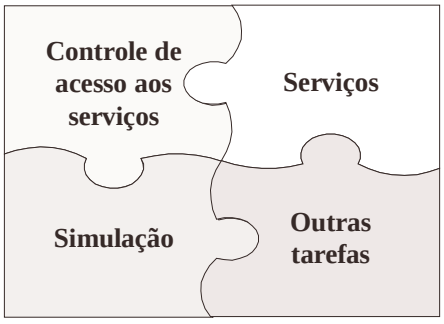


Figura 14. Diagrama da abordagem utilizada em aplicações monolíticas.

Esse tipo de abordagem foi inicialmente adotado na construção da primeira versão do sistema de realidade virtual militar NPSNET [Macedônia, 1995], mas não foi utilizado por muito tempo. Desenvolvido pela *Naval Postgraduate School*, atualmente esse sistema apresenta-se na sua quarta versão (Figura 15). A alta carga de processamento exigida por um sistema de simulação e o forte acoplamento dos serviços desse sistema comprometem seriamente o desempenho final de um sistema de realidade virtual. Assim, há necessidade de uma distribuição da carga de processamento entre várias máquinas em rede. Além da degradação do desempenho, a integração monolítica implica na reimplementação de todo o sistema sempre que for preciso incorporar novos serviços, o que constitui um laborioso processo.

Figura 15. Avatar de um soldado dentro do sistema NPSNET-IV.

3.2 Integração com ênfase no sistema de realidade virtual

Nessa abordagem de integração, o sistema de realidade virtual e a aplicação externa são dissociados, estando o controle do acesso aos serviços das aplicações externas a cargo do sistema de realidade virtual (Figura 16). Nesse caso, o ambiente virtual deve sofrer modificações de modo a permitir o gerenciamento e a manutenção da conexão com a aplicação externa, além de saber a linguagem entendida por essa aplicação para que as requisições realizadas dentro do ambiente virtual sejam atendidas corretamente.

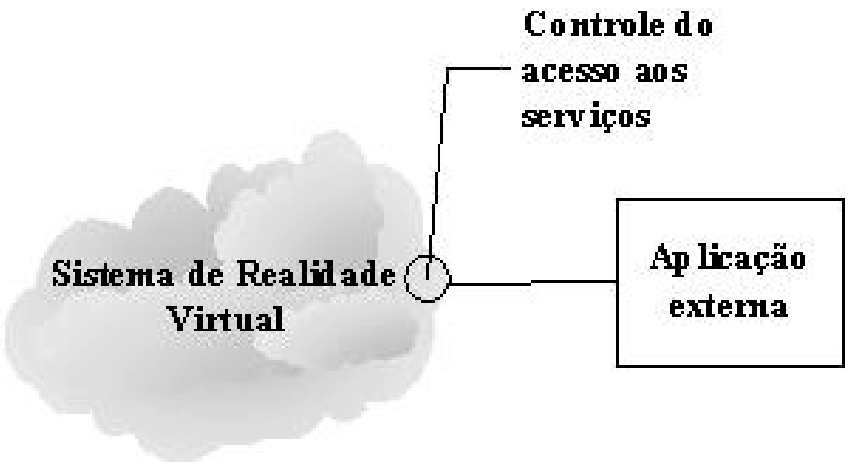


Figura 16. Diagrama da abordagem com ênfase no sistema de realidade virtual.

Exemplos desse tipo de abordagem são encontrados em sistemas de realidade virtual fechados (no sentido de que não fornecem uma interface aberta para comunicação externa) que, para fornecer serviços de outras aplicações, precisam ser modificados. É o caso do SIMNET (*Simulator Networking*), um ambiente virtual distribuído para treinamento de soldados do exército americano [Miller & Thorpe, 1995; Pope, 1989], e do BrickNet, uma ferramenta de autoria de ambientes virtuais que provê suporte para a modelagem dos aspectos gráficos, comportamentais e de rede de mundos virtuais [Singh et al., 1994]. Para permitir a utilização de ferramentas externas, a integração deve ser feita no momento da criação do ambiente virtual. O BrickNet permite que objetos sejam compartilhados por múltiplos mundos virtuais e tem sido usado para construir jogos em rede, sistemas de engenharia concorrentes, sistemas de apoio a atividades e formação de grupos (*groupware*), e outros ambientes gráficos assíncronos baseados em rede [Singh et al., 1995].

É importante ressaltar que a abordagem de integração utilizada por esses sistemas, caso haja necessidade de utilizar serviços de ferramentas externas, pode ser inferida a partir da análise de suas arquiteturas descritas em [Miller & Thorpe, 1995; Singh et al., 1994].

A grande vantagem dessa abordagem é possibilitar que as aplicações externas não sejam reescritas, ou seja, permite o aproveitamento de sistemas legados. Por outro lado, além das modificações do sistema do ambiente virtual, observa-se que se houver muitas requisições de vários usuários aos serviços dessa aplicação externa, o sistema do ambiente virtual pode ter o seu desempenho comprometido devido ao acúmulo de tarefas (tarefas pertinentes ao próprio sistema acrescidas do gerenciamento do acesso à ferramenta externa).

3.3 Integração com ênfase na aplicação externa

Ao contrário da abordagem anterior, nessa abordagem o controle de acesso aos serviços da aplicação externa é responsabilidade da própria aplicação externa (Figura 17). Para que isso ocorra, porém, é necessário que o sistema de realidade virtual forneça uma interface, ou seja, uma "porta aberta", para que outras aplicações possam conectar-se e tornar seus serviços acessíveis dentro do ambiente virtual.

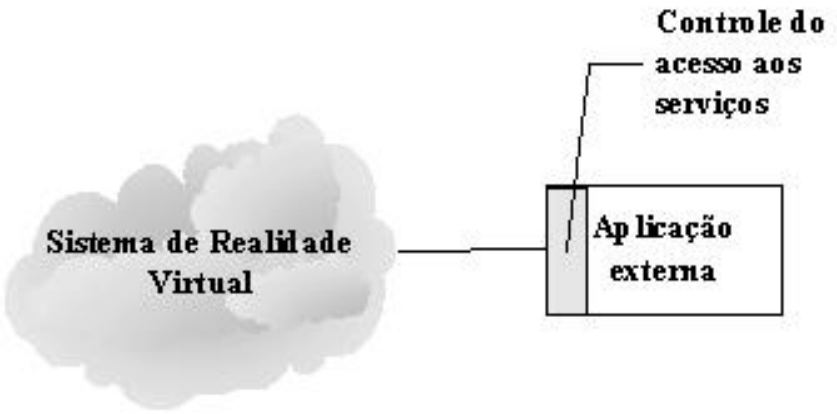


Figura 17. Diagrama da abordagem com ênfase na aplicação externa.

Nessa abordagem, a aplicação externa precisa estabelecer uma conexão com o sistema do ambiente virtual e entender o protocolo utilizado por esse sistema para a comunicação.

Essa interface também permite que aplicações controlem simulóides dentro desse ambiente, propiciando que esses simulóides apresentem comportamentos mais complexos [Pandzic et al., 1998]. Quanto mais simples for essa interface, mais fácil será a integração. O sistema VLNET (*Virtual Life Network*) é um exemplo de sistema de realidade virtual que fornece esse tipo de interface [Çapin et al., 1997; Pandzic et al., 1997]. O VLNET permite que um sistema de comportamento autônomo (aplicação externa) possa controlar um simulóide dentro do ambiente virtual, levando a uma espécie de simbiose: enquanto o sistema de comportamento autônomo fornece o cérebro, o sistema de realidade virtual fornece o corpo para que o simulóide movimente-se, seja visto e possa interagir com objetos, bem como ver, ouvir e obter as informações provenientes do ambiente para o cérebro (Figura 18).

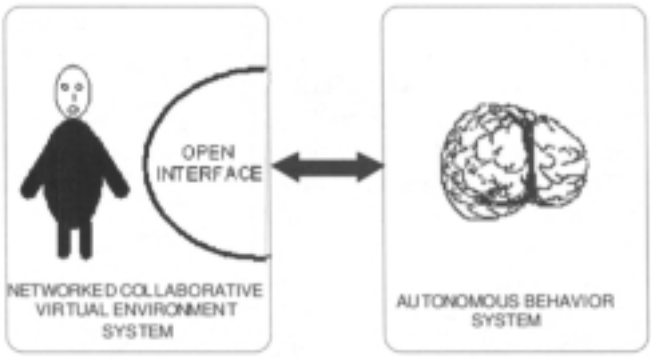
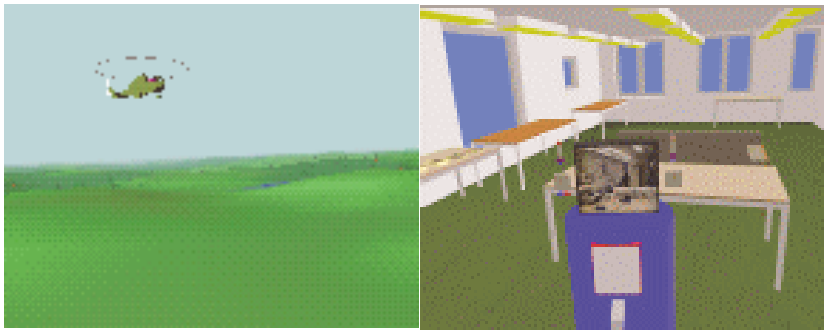


Figura 18. A simbiose entre o sistema de comportamento autônomo e o sistema de realidade virtual. Fonte: [Pandzic et al., 1998].

Através dessa abordagem, é possível manter vários simulóides com comportamentos diferentes dentro de um mesmo ambiente. Assim, conseguiu-se simular um jardim público onde ocorrem diversas interações sociais entre simulóides (Figura 19). Os simulóides dão uma volta pelo jardim e tentam estabelecer uma comunicação com outros simulóides. Cada simulóide pode possuir um estado emocional diferente, representado por suas posturas e seus gestos. Observando essa linguagem corporal, é possível inferir o tipo de relacionamento entre duas ou três pessoas e como a conversação está acontecendo mesmo que eles utilizem uma linguagem falada inaudível. Por exemplo, é possível saber até que ponto eles se conhecem bem e quão interessados estão na conversação que estão mantendo. Para essa simulação foi desenvolvido um modelo de comunicação não-verbal e de relacionamento interpessoal entre simulóides [Bécheiraz & Thalmann, 1996].

Figura 19. Simulóides interagindo em um jardim público virtual. Fonte: [Çapin et al., 1999].

Outro sistema acadêmico existente que se encaixa nessa abordagem é o DIVE (*Distributed Interactive Virtual Environment*), que consiste em um ambiente multiusuário baseado na internet, onde participantes navegam em um espaço tridimensional, vêem e interagem com outros participantes e aplicações [Frécon & Stenius, 1998]. As aplicações e as atividades DIVE incluem, entre outras, campos de batalha virtuais (Figura 20a), onde aplicações externas podem controlar tanques, helicópteros e outros veículos militares; e controle de robôs do mundo real (Figura 20b).



(a) (b)

Figura 20. (a) Módulo de terrenos realísticos. (b) Interface para robôs.

Na área de entretenimento, por sua vez, os jogos multiusuários baseados na interação em ambientes tridimensionais, tais como Doom e Quake (Figura 21), constituem importantes exemplos. Esses jogos são bastante difundidos entre os usuários de computador e permitem que os próprios usuários definam aplicações externas – os *bots*[1] (sistemas autônomos) – para assumirem o controle de personagens dentro do jogo.

Figura 21. Imagens capturadas dos jogos Doom II (esquerda) e Quake 3 (direita).

A principal vantagem dessa abordagem consiste no fato de o ambiente virtual não

[1] Termo derivado do inglês robot, largamente utilizado para definir os sistemas autônomos.

necessitar de modificações para permitir a conexão de aplicações externas. A desvantagem reside no fato de que nem sempre a reescrita de código das aplicações externas, de modo a controlar a disponibilização dos serviços dentro do ambiente virtual, é possível, principalmente em se tratando de sistemas legados sem acesso ao código fonte.

3.4 Integração através do uso de mediadores

Um mediador é um programa que coleta informações de uma ou mais fontes, processa-as, combina-as e repassa os resultados para outras aplicações [Wiederhold, 1992].

Existem vários tipos de mediadores, alguns mais simples tais com os *wrappers* [Ashish. & Knoblock, 1997; Carey et al., 1995; Franchitti & King, 1993], também conhecidos como tradutores [Papakonstantinou et al., 1995], que são responsáveis somente por converter os dados de cada fonte em um modelo de dados comum (Figura 22). Outros exemplos de mediadores mais complexos são os agentes inteligentes [Knapik & Johnson, 1998], aplicações autônomas capazes de procurar e filtrar informações, de adaptar-se em um ambiente dinâmico, e de desempenhar uma série de outras funções.

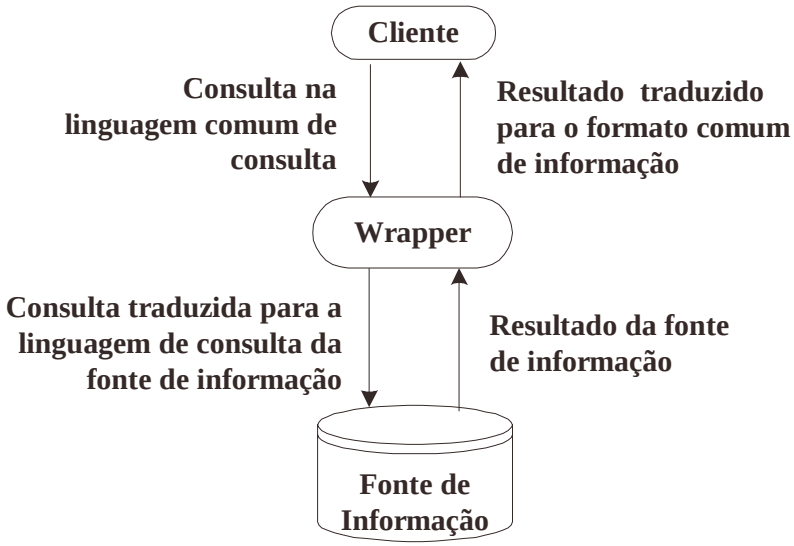


Figura 22. Comunicação através de um *wrapper* (tradutor). Fonte: [Papakonstantinou et al., 1995].

Os mediadores são largamente utilizados em arquiteturas de integração de sistemas heterogêneos em diferentes áreas da ciência da computação.

Na área de bancos de dados, principalmente enfatizando a integração de dados, existe a arquitetura TSIMMIS [Papakonstantinou et al., 1996]. Essa arquitetura (Figura 23) utiliza *wrappers* a fim de prover uma linguagem comum para extração de informação e

mediadores que combinam, integram ou refinam dados dos *wrappers* fornecendo uma visão “mais limpa” para as aplicações.

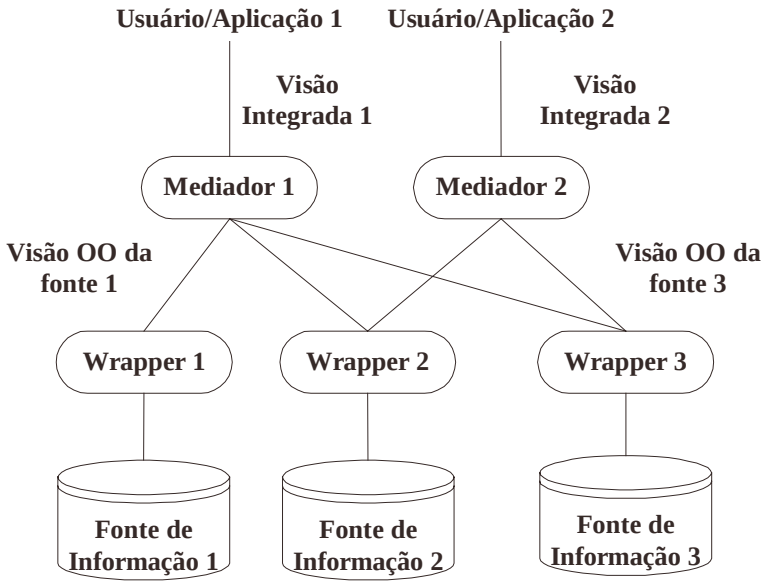


Figura 23. A arquitetura para integração de dados heterogêneos TSIMMS. Fonte: [Papakonstantinou et al., 1996].

Na área de engenharia de software, com domínio na integração de sistemas legados, [Souder. & Mancoridis, 1999] apresentam uma ferramenta que encapsula os serviços de uma aplicação legada redistribuindo-os como objetos distribuídos. A arquitetura dessa ferramenta é ilustrada na Figura 24.

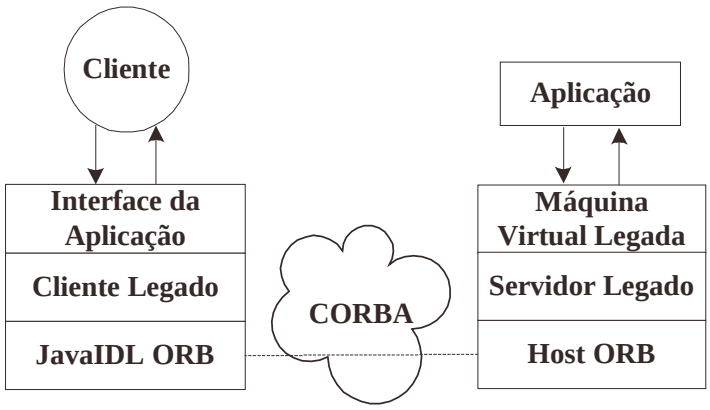


Figura 24. Modelo completo em camadas de um *wrapper* de sistemas legados. Fonte: [Souder. & Mancoridis, 1999].

Na área de inteligência artificial, por sua vez, principalmente na área de agentes inteligentes distribuídos [Bird, 1993; Genesereth, 1995; Knapik & Johnson, 1998], o problema maior é integrar agentes, de modo que se comuniquem e cooperem entre si.

[Genesereth, 1995] apresenta uma arquitetura para integração de agentes que utiliza facilitadores (Figura 25) para ajudarem na comunicação entre agentes escritos em diferentes plataformas.

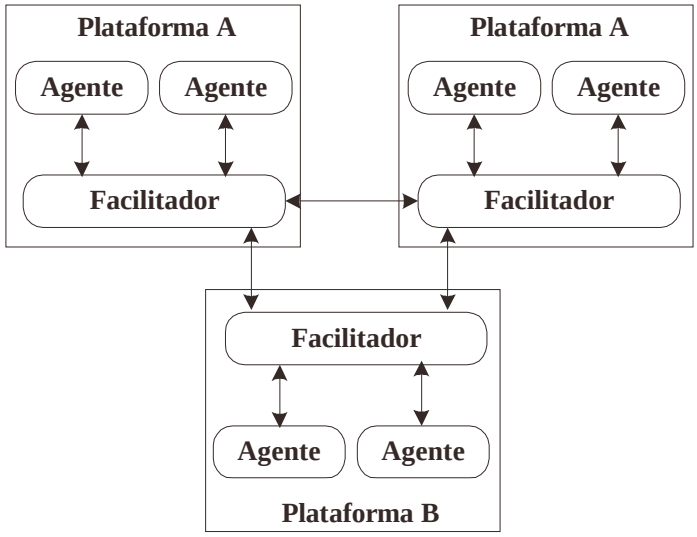


Figura 25. Arquitetura de interoperabilidade entre agentes. Fonte: [Genesereth, 1995].

Finalmente, na área de sistemas de informação [Mylopoulos et al., 1996; Wilderhold, 1992], a intenção é integrar diversos tipos de aplicações legadas, a exemplo da arquitetura CoopWARE, que integra diferentes tipos de serviços utilizando um conjunto de interfaces (Figura 26).

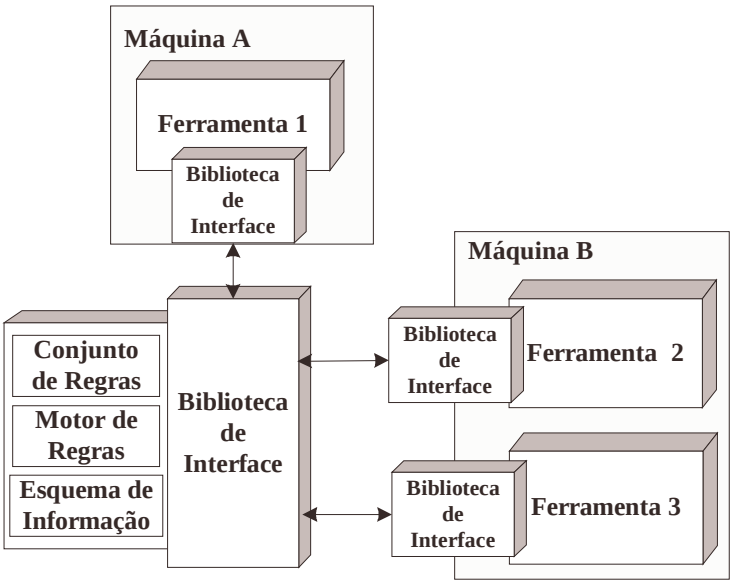


Figura 26. Arquitetura de integração de sistemas de informação CoopWARE. Fonte: [Mylopoulos et al., 1996]

Partindo desse princípio, percebe-se que os mediadores também podem ser aplicados para integrar aplicações externas a ambientes virtuais (Figura 27). A utilização de mediadores constitui uma solução eficiente porque ela permite que as duas extremidades ligadas pelo mediador permaneçam da forma que foram implementadas originalmente.

Figura 27. Diagrama da abordagem baseada no uso de mediadores.

Dessa forma, a aplicação mediadora assume as tarefas de controle de comunicação com as duas extremidades, além de gerenciar o acesso dos serviços disponibilizados dentro do ambiente virtual.

O conjunto dos sistemas que se enquadram nessa abordagem engloba todos os sistemas que se encaixam na abordagem anterior (integração com ênfase na aplicação externa), pois, se esses sistemas já possuem uma interface aberta para a conexão de uma aplicação externa, conseqüentemente permitirão a conexão com mediadores, bastando, para isso, que os mediadores entendam o protocolo de comunicação do ambiente virtual.

Um dos pontos mais interessantes em relação a um mediador é o modo como ele é definido internamente. Quanto mais modular for um mediador, ou seja, quanto maior for o número de módulos tratando tarefas específicas; mais fácil será de portá-lo para outros tipos de aplicações, já que só é preciso substituir os módulos necessários para atender as novas demandas [Mylopoulos et al., 1996].

O modelo de integração de aplicações externas a ambientes virtuais proposto nessa dissertação está baseado nessa abordagem e é descrito no capítulo a seguir.

Capítulo IV

VEGA – Modelo de Integração de Aplicações Externas a Ambientes Virtuais através de Aplicações Mediadoras e de Simulóides

O modelo proposto para integração do ambiente de realidade virtual a aplicações externas está dividido em três componentes básicos: o conjunto de aplicações externas, o simulóide e a aplicação mediadora. O esquema geral de relacionamento entre os componentes desse modelo encontra-se apresentado na Figura 28.

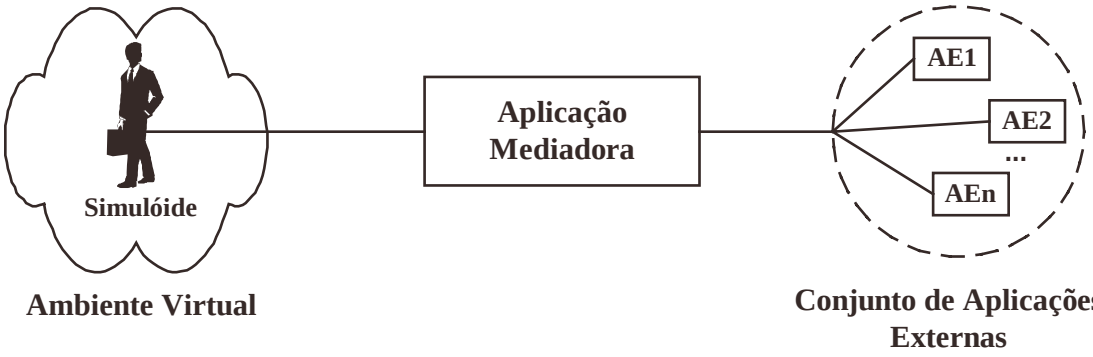


Figura 28. Esquema geral de integração.

O conjunto de aplicações externas consiste nas aplicações cujos serviços deverão ser disponibilizados aos usuários do ambiente virtual. O simulóide é o personagem virtual, controlado pela aplicação mediadora, que fornece os serviços de cada aplicação externa aos demais personagens virtuais do ambiente. A aplicação mediadora é a principal responsável pela integração propriamente dita entre a aplicação externa e o ambiente virtual.

Escolheu-se um simulóide como interface da aplicação por dois motivos principais. O primeiro é que o uso de simulóides em ambientes virtuais para desempenhar ações repetitivas pré-programadas ou para disponibilizar serviços especializados aos avatares ou a outros simulóides torna a experiência do usuário bem mais excitante e realista. Isso aprimora a capacidade de imersão do ambiente e conduz o usuário a um melhor sentimento de Presença [Batisti & Tarouco, 1999]. O segundo motivo refere-se ao fato de que o

projeto da grande maioria dos sistemas de realidade virtual distribuídos contempla uma interface aberta para a conexão de clientes. Assim o corpo do simulóide pode ser controlado por meio dessa interface. Ademais, não são oferecidas restrições quanto à utilização de interfaces mais simples, tais como as baseadas em menus, contanto que o sistema de realidade virtual dê suporte a esses outros tipos de interface.

O enfoque principal do modelo proposto para integração é a definição da estrutura computacional da aplicação mediadora. Dessa forma, o processo de incorporação de serviços ao ambiente torna-se bastante simplificado, sendo assim mais barato e mais eficiente do que portar serviços diretamente ao ambiente virtual ou à própria aplicação externa. A Figura 29 ilustra tanto o fluxo de mensagens que ocorre entre a aplicação mediadora e cada aplicação externa quanto o fluxo de mensagens entre a aplicação mediadora e o ambiente virtual. O funcionamento básico da aplicação mediadora resume-se a:

- traduzir solicitações realizadas pelos avatares do ambiente ao simulóide em requisições às aplicações externas;
- traduzir os retornos de cada aplicação externa em respostas aos avatares, através do simulóide;
- controlar o conjunto de ações do simulóide, representando o aspecto comportamental da interação com avatares.

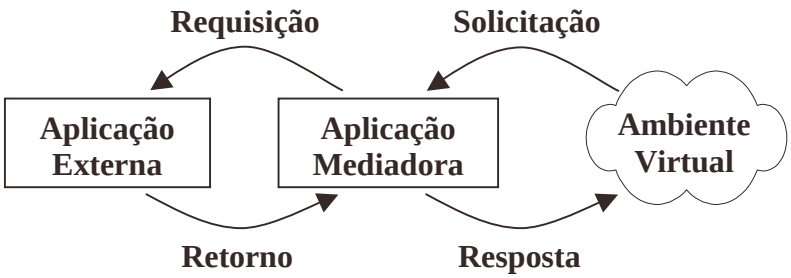


Figura 29. Fluxo de mensagens no modelo de integração proposto.

Os componentes básicos da aplicação mediadora são: tradutores das aplicações externas, tradutor do ambiente virtual, unidades de comunicação, gerente de comunicação, gerente de comportamento, gerente de conhecimento, unidades de conhecimento, gerente de simulóide e registrador de ocorrências. A Figura 30 apresenta como esses componentes estão relacionados dentro do Mediador. As responsabilidades de cada um desses componentes estão descritas a seguir.

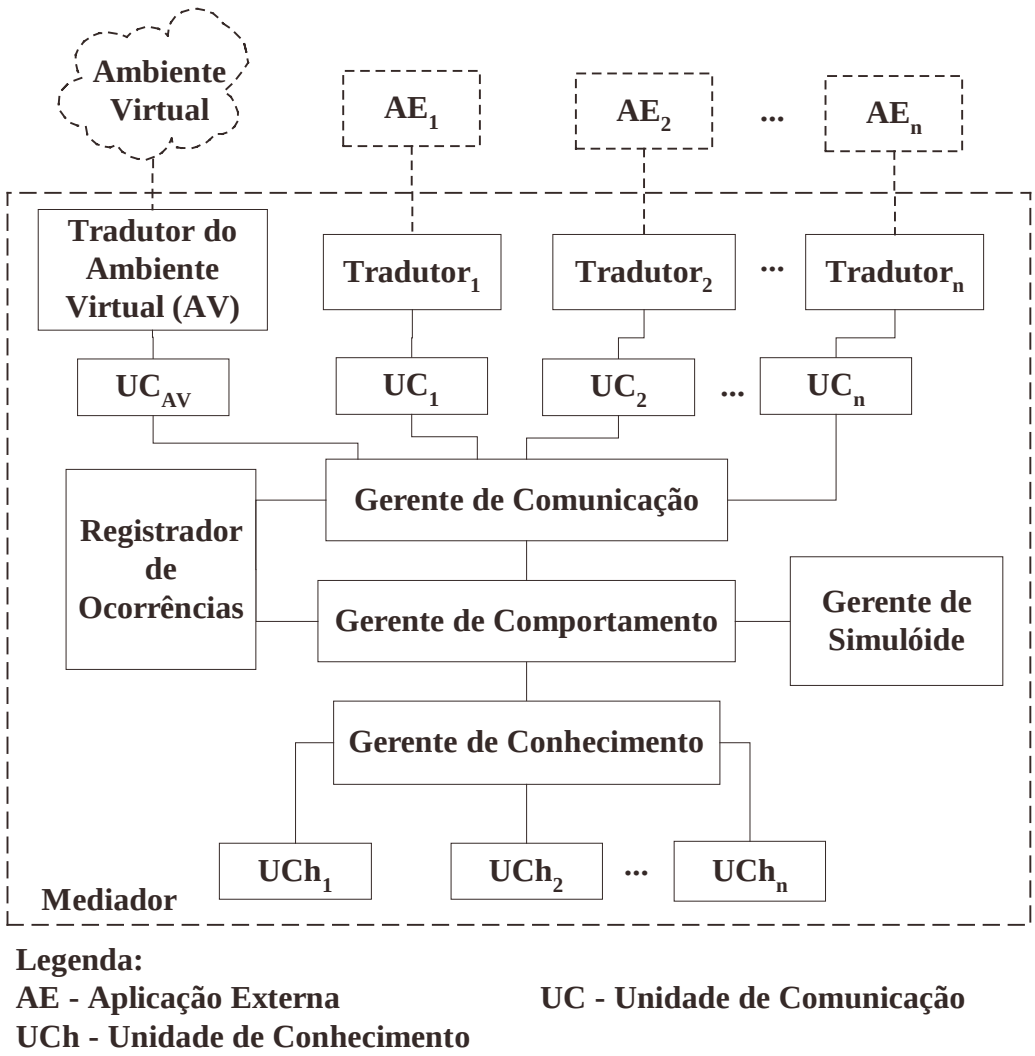


Figura 30. Diagrama de relacionamento entre componentes da aplicação mediadora.

4.1 Tradutores

Os tradutores são estruturas responsáveis pelo tratamento das informações transmitidas entre a aplicação mediadora e suas extremidades: a aplicação externa e o ambiente virtual. Para cada aplicação conectada ao mediador, existe um tradutor específico que traduz as mensagens oriundas de tal aplicação, repassando-as à sua respectiva unidade de comunicação, bem como converte as mensagens provenientes da sua unidade de comunicação e as envia para a sua extremidade, ambiente virtual (Figura 31a) ou aplicação externa (Figura 31b). Cada um dos tradutores da aplicação mediadora pode trabalhar com um protocolo de comunicação específico, de acordo com as características da extremidade à qual ele se liga. Esse tipo de comportamento dos tradutores está muito próximo daquele apresentado por um *proxy* no gerenciamento de redes [Stallings, 1997].

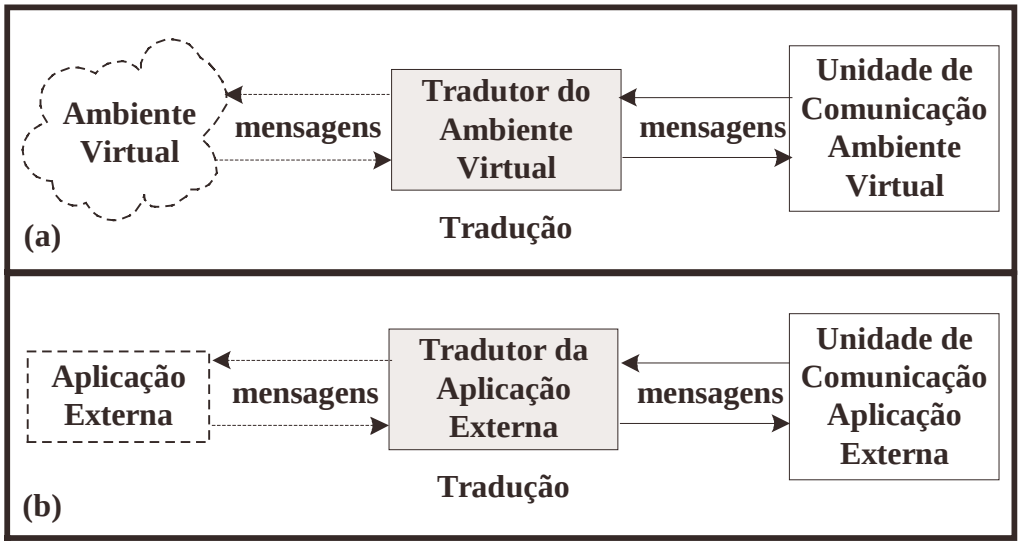


Figura 31.(a) Fluxo de mensagens no tradutor do ambiente virtual. (b) Fluxo de mensagens no tradutor da aplicação externa.

Os tradutores são estruturas de processamento bastante leves e fáceis de serem implementadas por serem específicos para cada protocolo de comunicação utilizado. Isso privilegia a manutenção posterior da aplicação mediadora, que poderá ser adaptada rapidamente a mudanças como, por exemplo, alterações no protocolo de comunicação de uma de suas extremidades.

4.2 Unidades de Comunicação

Uma unidade de comunicação é responsável por gerenciar o tradutor ligado diretamente a ela. Suas responsabilidades envolvem o estabelecimento e a manutenção das conexões de rede entre o tradutor, ao qual está ligada, e a extremidade ligada a esse tradutor; o repasse de mensagens recebidas do tradutor ao gerente de comunicação e o envio, ao tradutor, das mensagens recebidas do gerente de comunicação (Figura 32).

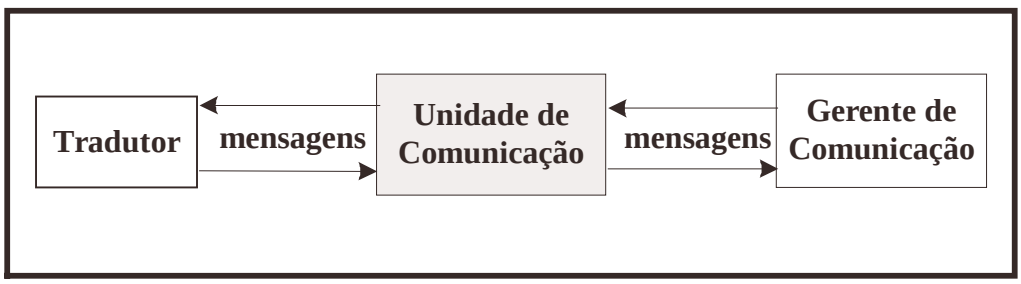


Figura 32. Fluxo de mensagens em uma unidade de comunicação.

Para os casos em que o ambiente virtual ou a aplicação externa realizam o controle de acesso a recursos, a unidade de comunicação também mantém as informações necessárias para validar a conexão entre a aplicação mediadora e sua extremidade. Por exemplo, em ambientes virtuais que exijam que seus usuários informem *login* e senha para se conectarem, a aplicação mediadora será responsável pelo cumprimento dessa exigência, através da unidade de comunicação ligada ao tradutor do ambiente virtual, a fim de permitir que o simulóide por ela controlado tenha acesso àquele ambiente virtual.

4.3 Gerente de Comunicação

O gerente de comunicação é o responsável pelo controle da comunicação entre os demais componentes da aplicação mediadora e suas extremidades. Ele funciona como interface entre todas as unidades de comunicação e o gerente de comportamento, repassando as mensagens recebidas das unidades de comunicação ao gerente de comportamento e enviando à unidade de comunicação específica, as mensagens provenientes do gerente de comportamento (Figura 33).

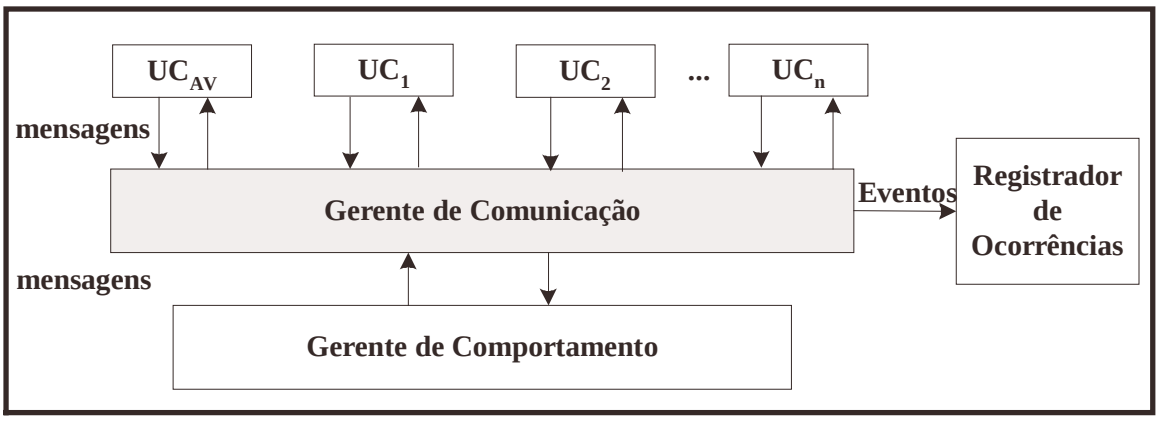


Figura 33. Fluxo de mensagens no gerente de comunicação.

É também responsabilidade do gerente de comunicação informar, quando requisitado, aos demais componentes do Mediador se todas as conexões com as extremidades estão ativas e funcionando.

Os eventos registrados pelo gerente de comunicação são automaticamente repassados ao registrador de ocorrências e referem-se ao funcionamento da capacidade de envio e recebimento de mensagens pela aplicação mediadora.

4.4 Gerente de Comportamento

O gerente de comportamento é o elemento central, responsável pelo tratamento das mensagens oriundas das aplicações externas e do ambiente virtual e recebidas através do gerente de comunicação (Figura 34).

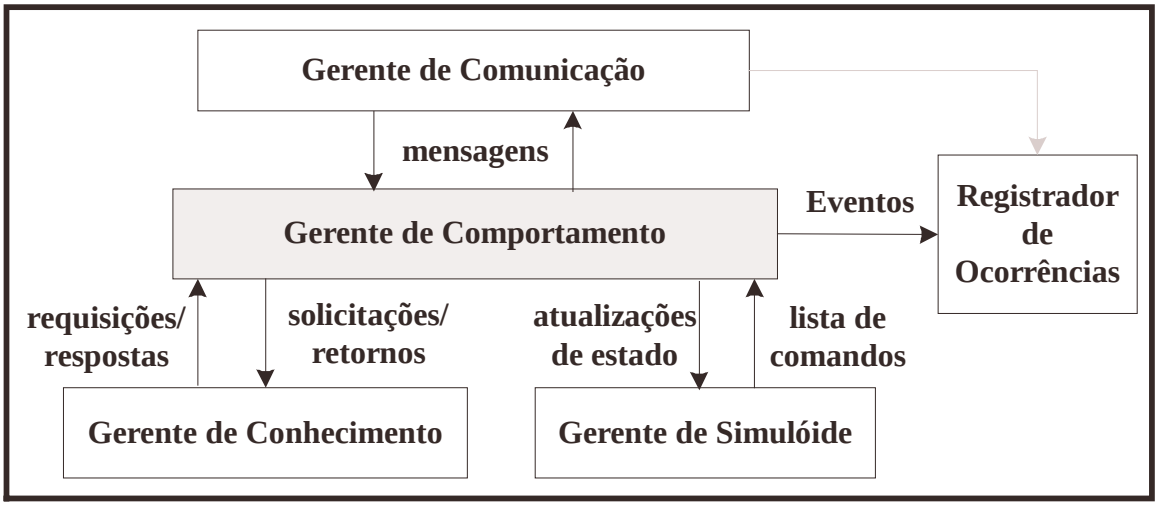


Figura 34. Detalhes do relacionamento entre o gerente de comportamento e os componentes ligados a ele.

O funcionamento genérico do gerente de comportamento está representado através de uma árvore de decisão na Figura 35.

As folhas das árvores representam as decisões a serem tomadas pelo gerente de comportamento ao receber uma mensagem. No primeiro nível, a decisão é baseada na origem das mensagens, que podem ser provenientes tanto da ferramenta externa, quanto do ambiente virtual, do gerente de conhecimento ou do gerente do simulóide. Quando as mensagens são recebidas a partir do gerente de conhecimento, é realizada uma segunda tomada de decisão, baseada no destino final das mesmas. O conjunto de possíveis ações a serem realizadas pelo gerente de comportamento, dependendo do resultado de suas decisões, é:

- repassar mensagem ao gerente de conhecimento, quando a mensagem recebida for uma solicitação do ambiente virtual ou um retorno da ferramenta externa;
- disponibilizar objetos de multimídia ao ambiente virtual;
- atualizar o estado do simulóide, enviando instruções ao gerente de simulóide;
- repassar mensagem diretamente para o gerente de comunicação, sempre que a mensagem recebida não precisar ser tratada.

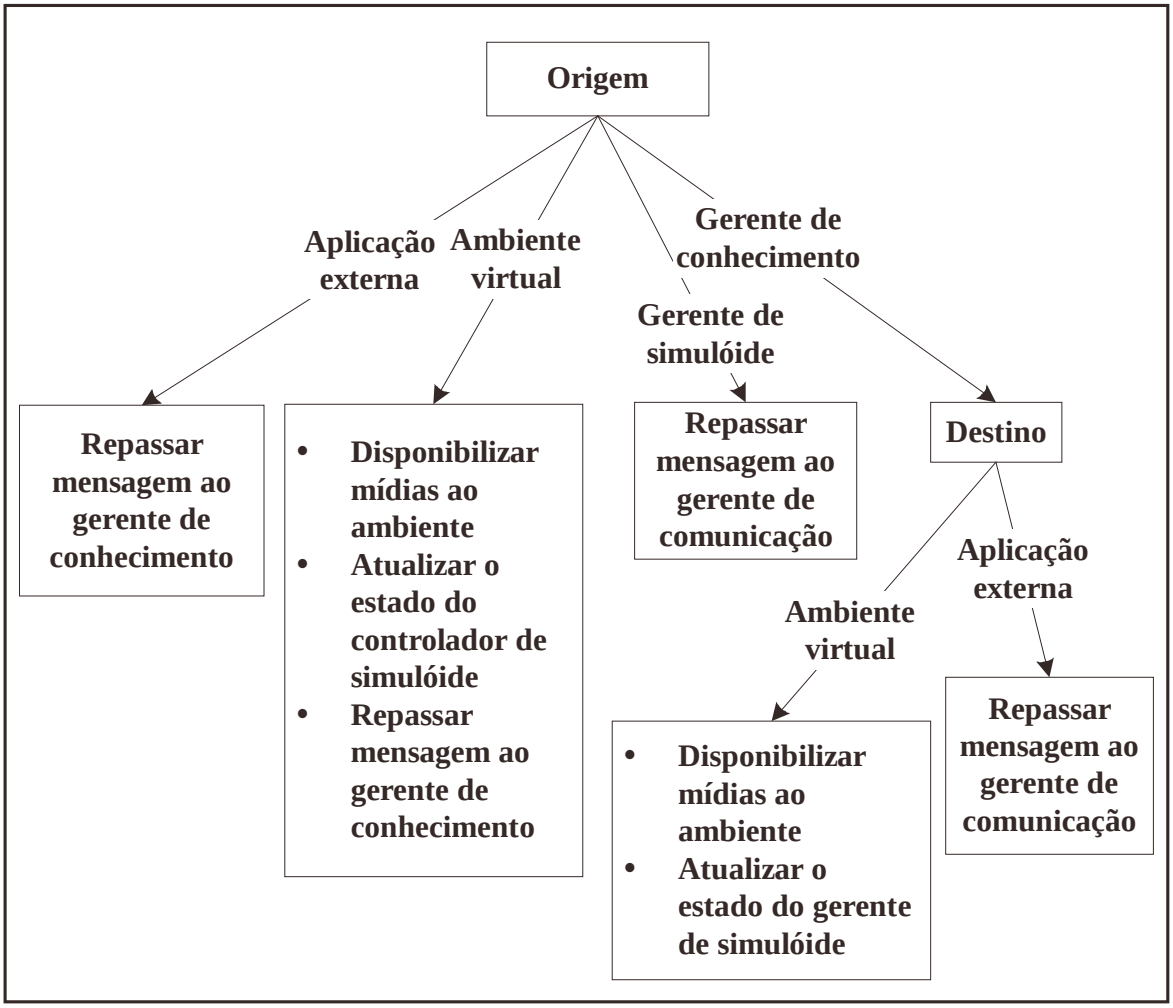


Figura 35. Árvore de decisões do gerente de comportamento.

A fim de registrar as tomadas de decisão e as conseqüentes ações realizadas pelo simulóide controlado pela aplicação mediadora, as ocorrências relacionadas às mensagens recebidas e enviadas pelo gerente de comportamento são repassadas e armazenadas no registrador de ocorrências.

4.5 Gerente de Conhecimento

O gerente de conhecimento faz a ligação entre o gerente de comportamento e as unidades de conhecimento (Figura 36). É sua responsabilidade identificar para qual unidade de conhecimento as solicitações ou os retornos devem ser encaminhados, assim como repassar as mensagens, já formatadas por cada unidade de conhecimento específica, para o gerente de comportamento.

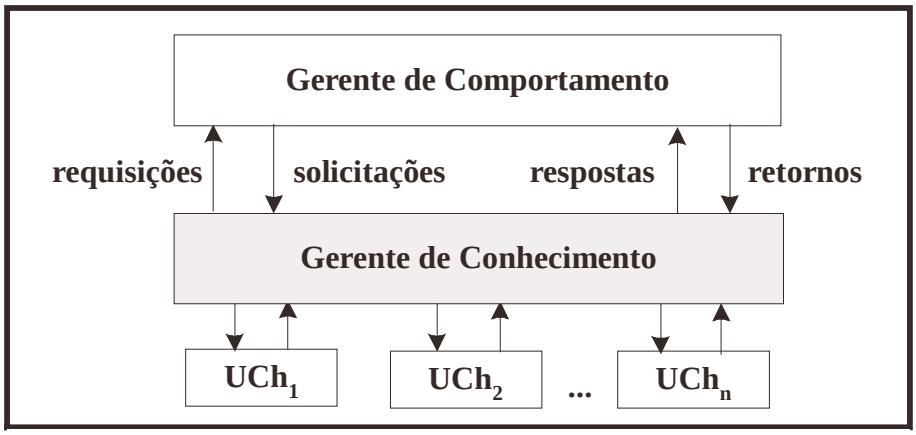


Figura 36. Fluxo de mensagens no gerente de conhecimento.

As mensagens seguem o caminho determinado até seu tradutor específico que se encarrega de enviá-las à aplicação externa.

4.6 Unidades de Conhecimento

Cada unidade de conhecimento é responsável por gerenciar informações de acesso a uma aplicação externa. Essas informações sobre o conjunto de serviços disponíveis referem-se a como tais serviços são requisitados e à semântica das informações retornadas por essas requisições. Portanto, para cada aplicação externa conectada ao mediador existe uma unidade de conhecimento específica.

Uma das responsabilidades de uma unidade de conhecimento é realizar o tratamento das solicitações provenientes do ambiente virtual (repassadas pelo gerente de conhecimento), gerando as respectivas requisições à aplicação externa. As unidades de conhecimento também são responsáveis pelo tratamento dos retornos da aplicação externa (Figura 37).

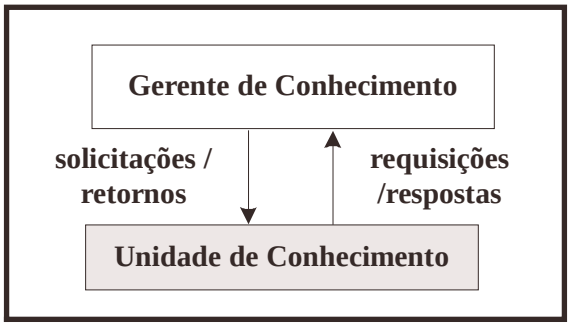


Figura 37. Fluxo de mensagens em uma unidade de conhecimento.

É importante ressaltar que, nos casos em que as solicitações oriundas do ambiente virtual forem suficientemente complexas, tais retornos podem resultar em novas requisições à própria aplicação externa relacionada a essa unidade de conhecimento ou a outras aplicações externas de outras unidades de conhecimento. Por exemplo, caso uma aplicação externa seja um gerenciador de banco de dados, uma solicitação originada no ambiente virtual pode ser convertida, pela unidade de conhecimento específica, em uma série de sub-consultas. O gerenciamento dessas consultas fica a cargo da própria unidade de conhecimento, que somente poderá repassar a resposta final ao gerente de conhecimento para que ele encaminhe ao ambiente virtual após a conclusão das mesmas.

4.7 Gerente de Simulóide

O gerente de simulóide é o componente responsável pela realização das mudanças de estado do simulóide no interior do ambiente virtual, a partir de instruções do gerente de comportamento. Para que isso seja possível, o gerente de simulóide mantém o estado atualizado do simulóide e um conjunto de possíveis ações, que relacionam as instruções de atualização de estado com conjuntos de comandos de animação (Figura 38). Esses comandos de animação podem englobar funções de deslocamento espacial, de movimentação facial e de outras funções, de acordo com os recursos de animações disponíveis no ambiente virtual. As mudanças de estado recebidas pelo gerente de simulóide são posteriormente repassadas ao simulóide, refletindo-as, então, no ambiente virtual.

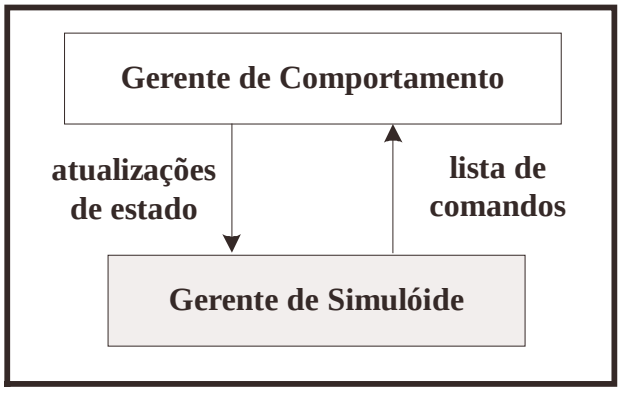


Figura 38. Fluxo de mensagens no gerente de simulóide.

4.8 Registrador de Ocorrências

O registrador de ocorrências é responsável pela documentação das informações provenientes do gerente de comunicação e do gerente de comportamento (Figura 39). As informações provenientes do gerente de comunicação relacionam-se aos dados sobre o fluxo de mensagens e aos procedimentos referentes às conexões com as extremidades da aplicação mediadora. Já as informações provenientes do gerente de comportamento dizem respeito às solicitações de usuários do ambiente virtual, às decisões tomadas para o atendimento às mesmas e aos demais processos relacionados. Essas informações podem ser usadas posteriormente para atividades como auditoria e administração do sistema.

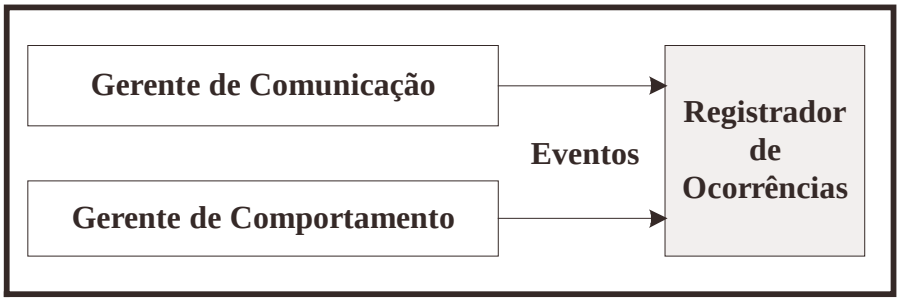


Figura 39. Fluxo de mensagens no registrador de ocorrências.

Capítulo V

Estudos de Casos do Modelo VEGA

A seguir são descritas duas aplicações baseadas no modelo proposto no capítulo anterior. A primeira aplicação, chamada BIA, integra dois serviços de dicionário eletrônico (o primeiro está localizado localmente e o outro está disponível na *web*) a um ambiente multiusuário de realidade virtual voltado para o treinamento de guias de turismo. A outra aplicação, denominada Beremiz[2], integra o software Mathematica a uma sala virtual de matemática.

Os ambientes virtuais dessas duas aplicações estão baseados na arquitetura Ataxia descrita em [Leite Jr, 2000]. Na Figura 40 está ilustrado como essa arquitetura é empregada para a disponibilização dos ambientes virtuais nos quais os simulóides estão inseridos.

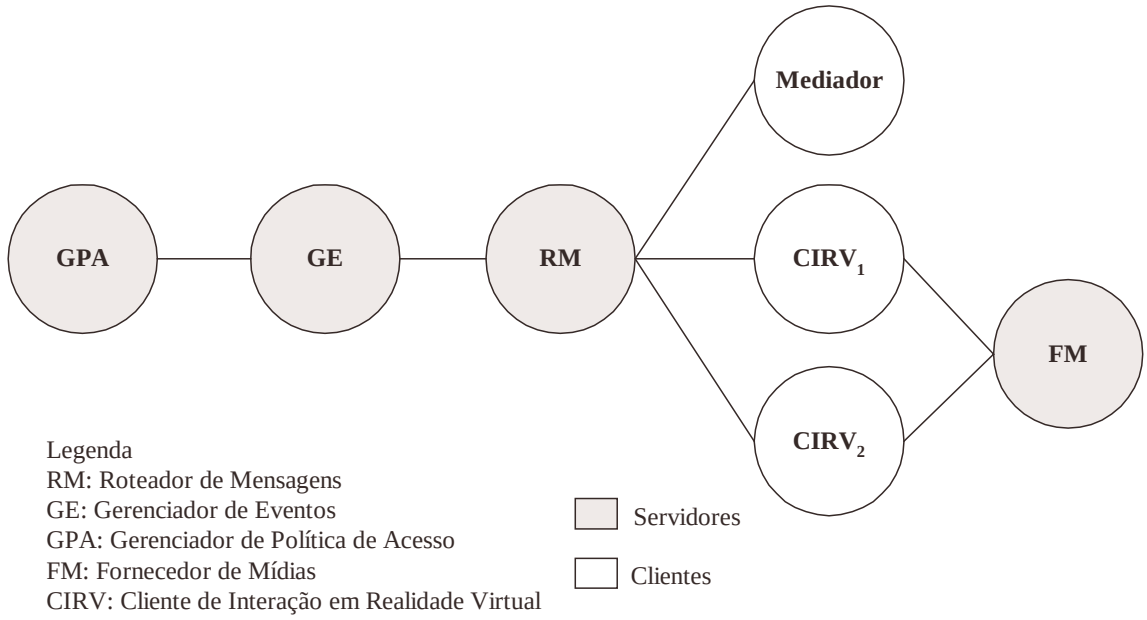


Figura 40. Emprego da arquitetura Ataxia na construção dos ambientes virtuais dos protótipos.

Baseando-se no paradigma cliente-servidor, a arquitetura Ataxia é formada de

[2] Referência a Beremiz Samir, famoso personagem da obra *O homem que calculava* do autor Malba Tahan.

componentes especializados responsáveis pela realização de tarefas específicas (gerenciamento de eventos, controle de acessos de participantes, etc.) e que podem ser distribuídos por uma rede de computadores comuns. Desse modo, os usuários fazem uso da aplicação CIRV – Cliente de Interação em Realidade Virtual para se conectarem a elementos específicos do sistema, tais como o Roteador de Mensagens e o Fornecedor de Mídias. O Mediador que controla um simulóide aproveita essa interface do Roteador de Mensagens, simulando a conexão de um cliente humano. Para o sistema como um todo, não há distinção se o cliente que está conectado é um simulóide ou um humano.

5.1 A Bibliotecária Virtual – BIA

A aplicação BIA foi desenvolvida baseada no modelo VEGA e integra os serviços de dois dicionários eletrônicos a um ambiente multiusuário de realidade virtual em rede voltado para o treinamento de guias de turismo [Vidal, 1999]. Nesse ambiente, os alunos são levados ao aprimoramento de seus conhecimentos de línguas, ao serem expostos a situações semelhantes às vivenciadas no mundo real. Alunos e instrutores, representados por seus avatares, podem participar do treinamento de maneira cooperativa, juntamente com simulóides inseridos no ambiente virtual. Os usuários podem acessar o ambiente virtual de treinamento tanto a partir de um mesmo laboratório, quanto de localidades remotas, através da Internet.

A fim de disponibilizar dentro do ambiente virtual um mecanismo de consulta a dicionário, foi desenvolvido um simulóide que personifica uma bibliotecária pronta a buscar informações que auxiliem a responder as questões de vocabulário de um usuário do ambiente. Essa bibliotecária, a BIA, encontra-se em uma biblioteca virtual à qual os usuários podem ser teletransportados instantaneamente sempre que precisarem consultar seus serviços (*virtual help*) (Figura 41). A BIA responde a solicitações dos usuários, informando as traduções de determinadas palavras. Para isso, ela utiliza os serviços de duas aplicações de dicionário eletrônico externas ao ambiente de realidade virtual. Uma dessas aplicações constitui um glossário bilíngüe desenvolvido especialmente para os usuários do curso de guias de turismo. A outra aplicação é o *Babylon* (http://www.babylon.com), um dicionário disponível na web.

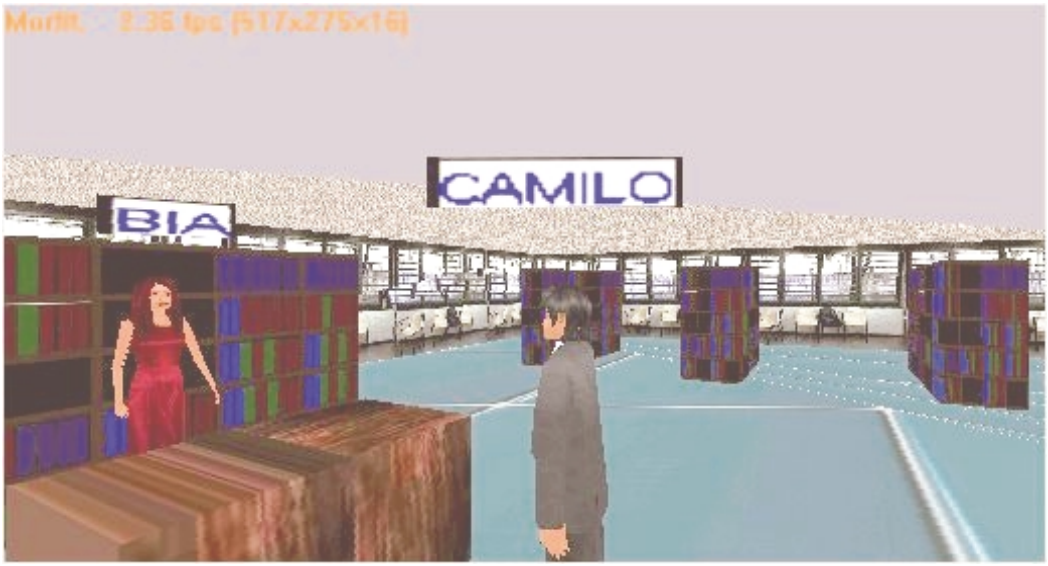


Figura 41. Avatar consultando a bibliotecária virtual.

5.1.1 Características da BIA

Sempre que o usuário aproxima-se da BIA, ela o saúda com um “bom dia!”, “boa tarde!” ou “boa noite!”, dependendo da hora vigente no ambiente virtual e se oferece para ajudar o usuário. A cada solicitação do usuário, através de mensagens escritas, ela apresenta uma resposta escrita, falada ou ambas.

Na Tabela 1, encontram-se os tipos de solicitações e as especificações dos formatos das respectivas sentenças.

Tabela 1. Formato de sentenças referentes às solicitações dos usuários.

Tipo de Solicitação	Formato da Sentença
Tradução português - inglês	How can I say <palavra> in English?
Tradução inglês - português	How can I say <palavra> in Portuguese?

Atualmente, o reconhecimento dessas sentenças é realizado através de simples rotinas de manipulação de cadeias de caracteres, de onde são extraídos os elementos necessários às solicitações. Visando uma maior flexibilidade dos formatos de requisições, um analisador léxico e sintático pode ser incorporado sem grandes dificuldades [Knapik & Johnson, 1998].

Para determinados tipos de situações, a BIA apresenta comportamentos distintos, como os mostrados na Tabela 2.

Tabela 2. Ações comportamentais da BIA.

Situação	Ação da BIA
Chegada de um novo usuário à biblioteca	Emitir saudação inicial.
Recebimento de solicitação de um usuário	Sorrir para o usuário e solicitar que ele aguarde a resposta de sua solicitação.
Retorno de uma resposta	Voltar-se para o usuário e informar o resultado da busca.
Não entendimento da sentença	Exibir uma mensagem padrão de sentença não compreendida.

5.1.2 Implementação da BIA

A aplicação foi implementada de forma a atender às especificações do curso para guias de turismo. De acordo com esse curso, quando o usuário realizar uma consulta a uma determinada palavra, é fundamental procurar inicialmente no glossário e, caso a consulta não obtenha sucesso, o *Babylon* será acessado. Tudo isso porque o glossário já está contextualizado para o conteúdo de cada lição do curso. Com esse objetivo, ao receber uma solicitação de um usuário, o mediador envia uma requisição ao glossário e se a palavra não for encontrada, o mediador faz uma requisição ao *Babylon* que retorna um arquivo HTML. Portanto, quando a consulta é bem sucedida, já na sua primeira etapa, o mediador envia uma mensagem de texto normal para o usuário. No entanto, quando a consulta não obtém sucesso na sua primeira etapa, o mediador envia para o usuário o arquivo HTML contendo os resultados da busca no *Babylon*, que podem ser positivos ou não.

O glossário local empregado foi desenvolvido em Delphi e disponibiliza suas funcionalidades através da rede, utilizando-se de mensagens transmitidas sobre o protocolo de comunicação TCP. O ambiente virtual em questão também faz uso desse protocolo para a conexão de seus usuários. O *Babylon* disponibiliza seus serviços através do protocolo HTTP. Com base no modelo de integração proposto, foi definida uma aplicação mediadora que se comunica diretamente com o dicionário eletrônico e com o ambiente virtual através do protocolo TCP e, com o *Babylon*, através do protocolo HTTP. Essa aplicação mediadora, desenvolvida em Delphi, também realiza o controle da BIA no interior do ambiente e disponibiliza as funcionalidades dos dicionários eletrônicos (Figura 42).

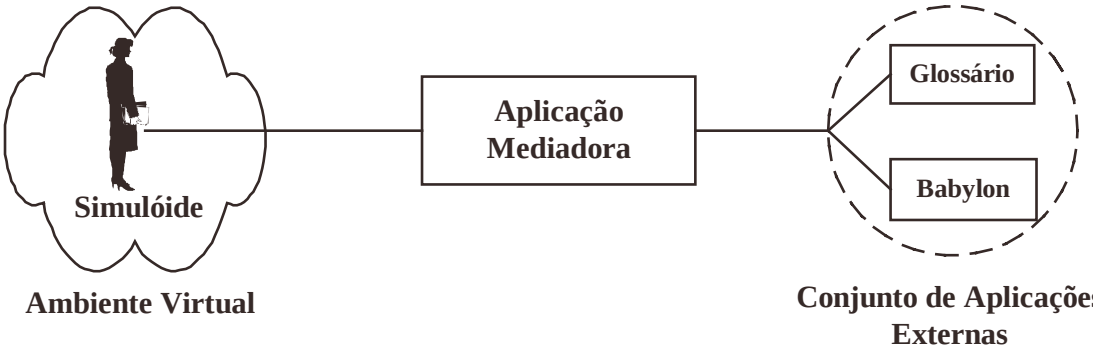


Figura 42. Modelo de implementação da BIA.

Aplicando o modelo VEGA a essa aplicação específica, obtém-se o diagrama da Figura 43.

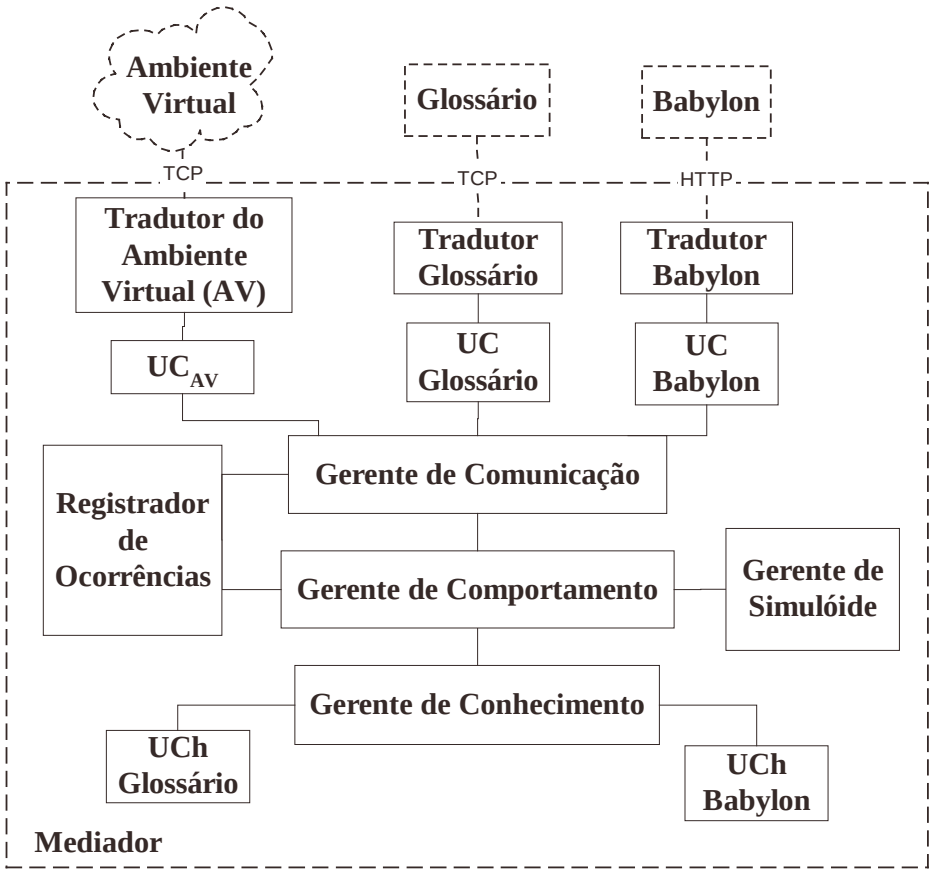


Figura 43. Modelo VEGA aplicado à BIA.

Considerando-se os requisitos de comunicação das aplicações externas utilizadas (o glossário e o *Babylon*) e do ambiente virtual ao qual a aplicação mediadora também se

conecta, foram implementados os seguintes componentes de comunicação específicos, definidos na aplicação mediadora.

Tradutores: dois tradutores que fazem uso direto do protocolo TCP, pois tanto o glossário quanto o ambiente virtual empregam esse protocolo para suas comunicações, e um tradutor que se comunica via HTTP. O funcionamento dos tradutores é bastante simples, realizando apenas conversões dos dados enviados e recebidos.

Unidades de comunicação: três unidades de comunicação responsáveis pelos processos básicos de conexão entre os tradutores e as respectivas extremidades. Como tanto o glossário utilizado quanto o ambiente virtual realizam a validação de acesso a seus recursos (*login* e senha), cada unidade de comunicação dessas extremidades também é responsável pelos processos de validação de acesso a cada sistema. Cada unidade reporta ao gerente de comunicação informações a serem registradas pelo registrador de ocorrências.

Gerente de comunicação: o gerente de comunicação deve encaminhar informações relevantes aos processos de conexão e validação de acesso para o registrador de ocorrências, incluindo tentativas de conexão em cada uma das extremidades do mediador (glossário, *Babylon* e ambiente virtual) e respectivos dados (dia, hora, ocorrência e possível causa) sobre sucessos e falhas. Além disso, o gerente de comunicação também informa ao gerente de comportamento se os tradutores estão conectados às suas extremidades.

De acordo com as necessidades da BIA, um conjunto básico de reações a situações geradas no ambiente virtual foi definido (Tabela 3).

Tabela 3. Reações da BIA referentes às situações do ambiente virtual.

Situação Gerada no Ambiente Virtual	**Reação da BIA**
Usuário entra na biblioteca virtual	1. Usar expressão facial “sorrindo”; 2. Consultar hora do ambiente virtual; 3. Apresentar som de “bom dia!”, “boa tarde!” ou “boa noite!”, de acordo com a hora do ambiente virtual; 4. Exibir mensagem de “posso ajudá-lo?”; 5. Usar expressão facial “normal”.

Solicitação de tradução por um usuário	1. Usar expressão facial “sorrindo”; 2. Exibir mensagem de “aguardar” para o usuário; 3. Montar, respectivamente, “tradução para o português” ou “tradução para o inglês”; 4. Enviar “tradução para o português” ou “tradução para o inglês” ao glossário; 5. Aguardar resultado do glossário; 6. Extrair dados da resposta do dicionário eletrônico; 7. No caso de insucesso: a. Enviar mensagem para o usuário avisando que não encontrou a palavra solicitada no glossário e que está buscando na *web*; b. Montar, respectivamente, “tradução para o português” ou “tradução para o inglês”; c. Enviar “tradução para o português” ou “tradução para o inglês” ao *Babylon*; d. Aguardar resultado do *Babylon*; e. Montar arquivo HTML a ser enviado ao usuário; 8. Virar a BIA para o usuário; 9. Usar a expressão de “sorrindo”; 10. Apresentar, respectivamente, som de “busca realizada”; 11. Exibir mensagem (texto ou arquivo HTML) contendo resultados da consulta ao usuário; 12. Usar expressão facial “normal”.
Requisição do usuário não compreendida	1. Usar expressão facial “desconsertado”; 2. Exibir mensagem de “não compreendi”; 3. Apresentar som de “perdão!”; 4. Usar expressão facial “normal”.

A fim de se gerenciar essas reações e com base no modelo VEGA, os seguintes componentes foram implementados.

Gerente de comportamento: a interação dos demais usuários do ambiente virtual com a BIA origina solicitações que são repassadas ao gerente de comportamento através dos componentes específicos de comunicação, explicitados anteriormente. Parte dessas solicitações constitui apenas mensagens de movimentação as quais são analisadas e, de acordo com essa análise, o gerente de comportamento coordena as possíveis reações da BIA. Em casos específicos de animação da BIA, o gerente de comportamento envia ordens específicas de mudanças de estado ao gerente de simulóide, a fim de que a personificação virtual, no interior do ambiente virtual, possa movimentar-se, modificar suas expressões faciais, etc. Para enriquecer as interações da BIA com os usuários do ambiente virtual, mídias auxiliares podem ser disponibilizadas a partir de um banco de dados de mídias, ligado diretamente ao gerente de comportamento, e repassadas ao ambiente virtual. A outra

parte das solicitações exige um tratamento mais aprofundado. Nesses casos, o gerente de comportamento repassa as mensagens ao gerente de conhecimento.

Gerente de conhecimento: as mensagens oriundas do ambiente virtual são repassadas do gerente de comportamento ao gerente de conhecimento. Essas solicitações são, então, identificadas e repassadas à unidade de conhecimento responsável por tratar aquela solicitação. O gerente de conhecimento também recebe os retornos das aplicações externas, encarregando-se de identificar o tipo de dado retornado e repassá-lo à respectiva unidade de conhecimento.

Unidades de conhecimento: Duas unidades de conhecimento foram implementadas. Cada uma delas é responsável por gerar as requisições no formato utilizado pela sua aplicação externa. Caso a consulta ao glossário não obtenha sucesso, a unidade de conhecimento do glossário deve enviar os dados solicitados de volta ao gerente de conhecimento, que por sua vez, repassa-os à unidade de conhecimento do *Babylon*. Dessa forma, a requisição é montada no formato compreendido pelo próprio *Babylon*. O resumo do tratamento realizado pelas unidades de conhecimento e as mensagens repassadas ao gerente de conhecimento, de acordo com cada tipo de retorno das aplicações externas, podem ser vistos na Tabela 4. O diagrama de seqüência referente a essas situações encontra-se detalhado no Anexo 3.

Tabela 4. Mensagens geradas pelo Mediador ao usuário, de acordo com os tipos de retornos do glossário.

Tipo de Retorno do Dicionário Eletrônico	Tipo de Mensagem Gerada	Conteúdo da Mensagem Gerada
Palavra encontrada no glossário	Exibição de texto	Lista com uma ou mais descrições oriundas do glossário.
Palavra não encontrada no glossário	Exibição de arquivo HTML	Resultados (positivos ou negativos) da busca no dicionário da *web*

Gerente de simulóide: com o intuito de tornar as reações da BIA mais realistas, o gerente de simulóide é responsável pelo controle das funções de posicionamento, de animação e de expressão facial específicas da BIA no interior do ambiente virtual. Sua construção é bastante simples, consistindo basicamente de um conjunto de regras de comportamento implementado na forma de um arquivo de *scripts*, que estabelecem a relação entre ações comportamentais e conjuntos de funções de posicionamento, de animação e de expressão facial. Um trecho desse arquivo é exibido na Figura 44. O arquivo completo encontra-se no Anexo 4.

```
;; Arquivo de inicialização do controle de simulóide da BIA
;; Os comandos devem ser colocados na ordem em que eles
;; serão executados. Posteriormente colocar o tipo
;; de mensagem a que eles correspondem
;; O primeiro caractere depois do = definirá o tipo:
;; T: mensagem de texto
;; M: movimentação do simulóide (ação)
;; S: som
;; E: expressão facial
;; D: deslocamento

;Saudação Inicial Manhã
[SIM]
0=D,Virar para usuário
1=E,sorrindo
2=T,Good Morning!
3=S,bomdia.mp3
4=T,May I help you?
5=E,normal

;Saudação Inicial Tarde
[SIT]
0=D,Virar para usuário
1=E,sorrindo
2=T,Good Afternoon!
3=S,boatarde.mp3
4=T,May I help you?
5=E,normal

;Saudação Inicial Noite
[SIN]
0=D,Virar para usuário
1=E,sorrindo
2=T,Good Evening!
3=S,boanoite.mp3
4=T,May I help you?
5=E,normal
```

Figura 44. Trecho do arquivo de *scripts* contendo os comandos para a atualização do estado do simulóide da BIA.

Registrador de ocorrências: informações gerais sobre as conexões mantidas entre o mediador e suas extremidades (ambiente virtual e dicionários eletrônicos) são armazenadas no registrador de ocorrências, em um simples arquivo de texto, incluindo data, hora e tipo de ocorrência (conexão realizada com sucesso ou falha). Uma outra função do registrador de ocorrências é a documentação das interações dos demais usuários do ambiente virtual com a BIA, incluindo data, hora, usuário do ambiente virtual que originou o evento, os tipos de requisições aos dicionários eletrônicos (tradução para o português ou para o inglês, e respectivos dados) e reações da BIA. Esse componente é bastante importante, pois

permite ao administrador do sistema avaliar, através de análise do arquivo gerado, o funcionamento da aplicação mediadora como um todo, acompanhar o comportamento da BIA e corrigir, caso seja necessário, alguma falha.

O completo diagrama de classes do mediador está ilustrado no Anexo 2. A fim de exemplificar o uso da BIA, a Figura 45 mostra uma situação no interior da biblioteca virtual na qual o usuário *camilo* interage com a BIA, que se encontra atrás do balcão. Ele está interessado em saber a tradução da palavra *car* para o português. Como a palavra *car* não está presente no glossário, a página *web* é, então, exibida. Todas as interfaces da aplicação BIA estão ilustradas no Anexo 1.

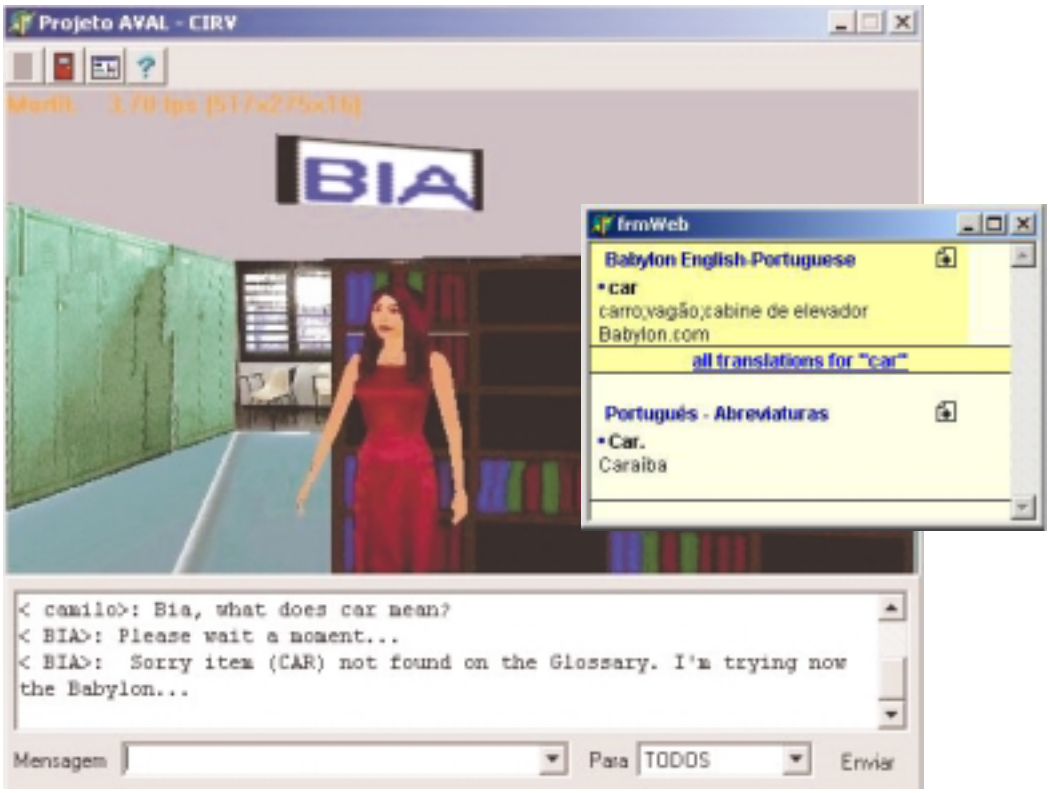


Figura 45. Tela do usuário Camilo fazendo uma requisição à BIA.

Uma versão da BIA que integra apenas um dicionário local, mas que executa operações de atualização no banco de dados é apresentada em [Vidal et al., 2000].

5.2 Monitor de Matemática - Beremiz

Nessa seção é descrita uma aplicação que integra os serviços do software Mathematica a uma sala virtual de matemática [Marques et al., 2001]. Essa aplicação controla o simulóide chamado Beremiz, monitor de Matemática em tempo integral (Figura 46).

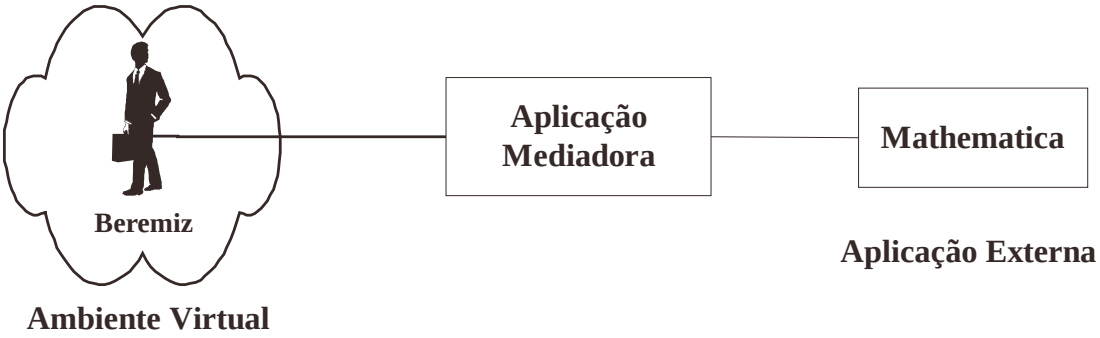


Figura 46. Esquema geral de integração do Mathematica a um ambiente virtual.

Antes de serem exibidas as características dessa aplicação, é importante explicar tantos as características do software Mathematica quanto as características da sala virtual de matemática para que a integração seja, então, bem compreendida.

5.2.1 A sala virtual de matemática

Ambientes virtuais compartilhados voltados para educação a distância possibilitam o desenvolvimento tanto de atividades educacionais comumente realizadas em escolas do mundo real quanto de atividades em ambientes imaginários ou que representem abstrações possíveis do mundo real.

Em um sub-ambiente virtual para realização de atividades voltadas para o aprendizado de matemática, é necessário que estejam disponíveis uma série de recursos que auxiliem essas atividades. O ambiente deve, por exemplo, fornecer os seguintes recursos:

- identificação dos usuários - feita através de avatares identificados por pseudônimos para permitir a interação no grupo, reforçando o sentimento de presença no mundo virtual;
- mecanismos de navegação - permitem a movimentação dos usuários dentro do ambiente;
- interação com objetos - os participantes do ambiente podem manipular objetos do mundo virtual, respeitando as características próprias de cada objeto;
- mecanismos de comunicação - permitem a troca de mensagens entre os usuários;
- quadro branco: permite o compartilhamento de uma área gráfica no interior do ambiente virtual, onde se pode inserir e manipular, de forma cooperativa, imagens, objetos gráficos e textos;

- projetor virtual de slides: permite que apresentações de conteúdo pré-desenvolvidas sejam expostas aos membros do grupo.

Além desses recursos há necessidade de utilizar, no ambiente virtual, os serviços de software largamente utilizados como ferramentas de apoio às atividades de ensino de matemática.

A sala virtual de matemática utilizada nessa aplicação dá suporte a todos os recursos descritos anteriormente e adota o paradigma cliente-servidor, onde clientes e servidores comunicam-se através do protocolo TCP.

5.2.2 O software Mathematica

O Mathematica constitui um ambiente completamente integrado capaz de lidar com os vários aspectos da computação técnica de um modo unificado e coerente. Tudo isso graças à idéia de utilizar uma linguagem de computação simbólica que permite manipular uma grande variedade de objetos envolvidos em computação técnica usando apenas um pequeno número de primitivas básicas.

O Mathematica tem sido largamente utilizado em educação [Wolfram, 2001], inclusive em áreas tais como química [Kent & Stevenson, 1999], ciências médicas, física, engenharia e estatística. Inicialmente utilizado apenas nas universidades e centros de pesquisa, tornou-se uma importante ferramenta tanto para estudantes de cursos técnicos como para estudantes de outros tipos de cursos não-técnicos.

A estrutura do Mathematica

O Mathematica é um software modular baseado no modelo cliente-servidor. O módulo chamado *kernel*, encarregado de efetuar cálculos computacionais, é separado do módulo *front end* que é responsável pela interação propriamente dita com o usuário.

As três formas de interação com o Mathematica são: através da utilização dos *Notebooks*; através da interface baseada em texto do próprio *kernel* e através do MathLink [Wolfram, 1996]. O MathLink permite ao Mathematica interagir não só com humanos como também com outros programas. Isso é possível porque o MathLink é um protocolo padronizado para a comunicação bidirecional entre programas externos e o *kernel* do Mathematica. Assim, a utilização do MathLink é o meio mais apropriado de integrar o Mathematica com o ambiente virtual, que será descrito posteriormente.

A biblioteca MathLink

MathLink é uma biblioteca de funções que implementa um protocolo para o envio e recebimento de expressões utilizadas pelo Mathematica. Uma característica importante dessa biblioteca é a de ser independente de plataforma, podendo usar qualquer mecanismo de comunicação entre programas. A conexão entre o *kernel* e o programa externo tanto pode ser efetuada em um mesmo computador como através de uma rede heterogênea de computadores.

O modo mais fácil e comum de se utilizar a MathLink ocorre quando o Mathematica realiza chamadas a funções externas, escritas em qualquer linguagem de programação. Assim, é relativamente fácil incorporar rotinas no Mathematica no caso de um algoritmo precisar ser implementado em uma determinada linguagem compilada por motivos de eficiência, ou no caso de simplesmente evitar-se a reescrita do código original em Mathematica.

A biblioteca MathLink também é utilizada quando um programa externo precisa fazer chamadas ao *kernel* do Mathematica, utilizando-o como uma ferramenta computacional. O próprio *front end* padrão do Mathematica comunica-se com o *kernel* via MathLink. É possível uma aplicação externa estabelecer uma conexão com o *kernel* simplesmente fazendo chamadas a funções da biblioteca. Nesse caso o *kernel* é executado no chamado Modo MathLink, significando que uma conexão é estabelecida com o programa externo de forma que qualquer entrada para o *kernel* virá desse programa e toda saída do *kernel* será direcionada para ele. Para tanto, é necessário saber: o formato apropriado de enviar expressões para o *kernel*, os tipos de resultados esperados no retorno e como obter esses resultados provenientes da conexão.

5.2.2 Características do Beremiz

O simulóide Beremiz, presente em uma sala virtual de matemática, disponibiliza os serviços do Mathematica para os usuários do ambiente da escola virtual. Portanto, o Beremiz pode solucionar equações, desenhar gráficos de funções no plano tridimensional ou bidimensional, guardar definições de varáveis, e realizar diversas outras tarefas que o Mathematica é capaz executar.

O modelo adotado para o desenvolvimento do Beremiz é uma simplificação do modelo VEGA, pois o mediador dessa aplicação integra apenas uma aplicação externa e também devido às próprias particularidades do Mathematica. As características da aplicação Beremiz são discutidas a seguir.

Tradutor da ferramenta externa disjunto da Aplicação Mediadora

Essa característica permite a execução da aplicação mediadora em uma máquina diferente daquela onde está situado o Mathematica, sem perda de controle ou dependência de plataforma. Quando o mediador e o *kernel* do Mathematica encontram-se em máquinas diferentes, existem duas maneiras do mediador acessar o *kernel* do Mathematica. Na primeira, o *kernel*, que está executando, espera uma conexão com o mediador. No entanto, isso gera a necessidade de intervenção do usuário para iniciar o *kernel* sempre que ocorrer uma falha. Para evitar essa intervenção indesejada, uma outra maneira consiste em instalar um serviço na máquina onde está o *kernel*, que se encarrega de iniciá-lo automaticamente sempre que necessário.

Desacoplando-se o tradutor do Mathematica da aplicação mediadora e instalando-o na máquina onde o *kernel* reside, pode-se dotar o tradutor da capacidade de iniciar o *kernel* e a biblioteca MathLink a partir de uma requisição remota por parte do mediador, sempre que necessário.

Com o objetivo de tornar a comunicação entre o tradutor do Mathematica e a aplicação mediadora mais direta, foi criado um protocolo específico de comunicação orientado às rotinas da biblioteca MathLink. Esse protocolo, chamado ProtoMath, é encapsulado em um componente a fim de tornar a programação mais fácil e modular. Dessa forma, a aplicação mediadora trata as rotinas do MathLink como se fossem locais.

Transferência de gráficos via FTP

Quando um participante do ambiente virtual solicita que o Beremiz mostre o gráfico de uma função (Figura 47a), o processo de transferência da imagem, via FTP, é realizado de forma transparente para o usuário e alguns segundos depois o gráfico é exibido em um Quadro Branco da sala de Matemática. Atendendo à solicitação do usuário, o Mathematica gera e envia para o Tradutor uma imagem no formato wmf (*Windows Metafile Format*) (Figura 47b). O Tradutor, então, converte a imagem para o formato jpeg (*Joint Photographic Experts Group*), mais compacto do que o formato wmf, e a envia para um servidor de FTP previamente conhecido por todos os clientes do ambiente virtual (Figura 47c).

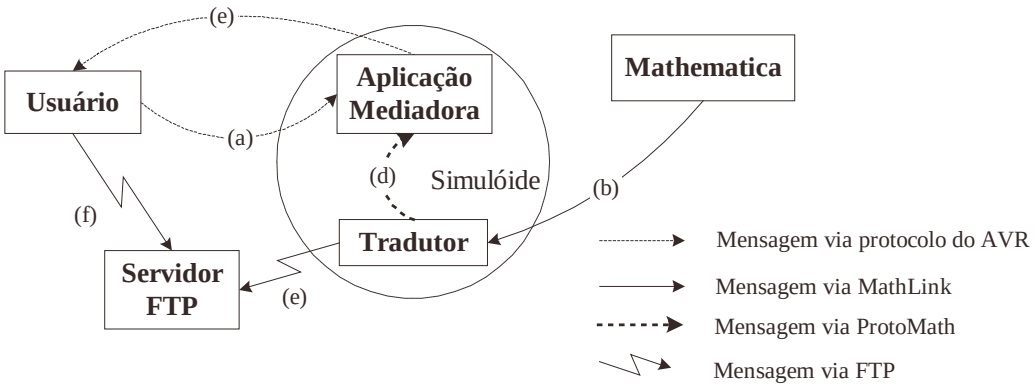


Figura 47. Esquema de requisição de gráficos.

Depois disso, o Tradutor envia uma mensagem para a aplicação mediadora informando que o arquivo já foi encaminhado para o servidor de FTP (Figura 47d). A aplicação mediadora, então, manda uma mensagem para o participante que fez a requisição, contendo tanto o nome do arquivo gráfico, quanto em qual Quadro Branco a imagem deve ser exibida (Figura 47e). A partir de então, a aplicação cliente do participante faz o *download* do arquivo diretamente do servidor de FTP (Figura 47f) e o mostra no Quadro Branco indicado (Figura 48).

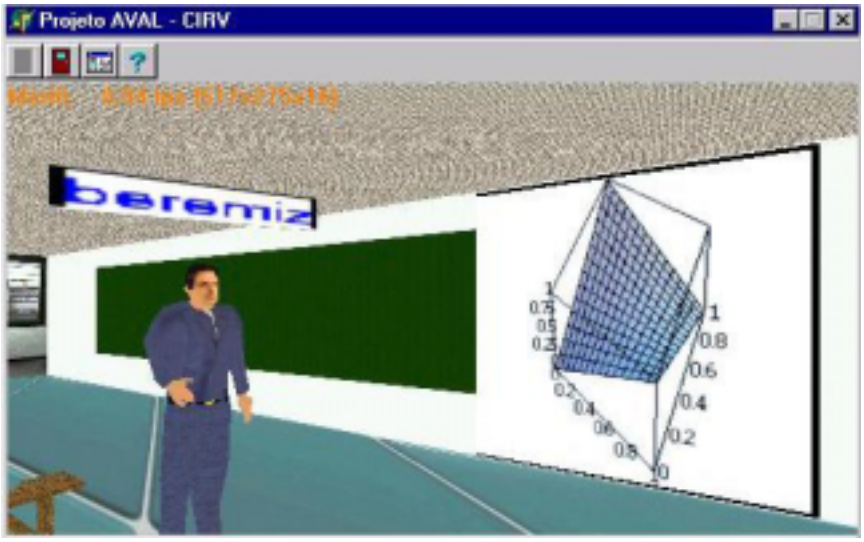


Figura 48. Usuário observando um gráfico desenhado no Quadro Branco.

5.2.3 Implementação do Beremiz

Devido ao fato de integrar ao ambiente virtual apenas uma aplicação externa, optou-se por uma simplificação do modelo VEGA ilustrada na Figura 49.

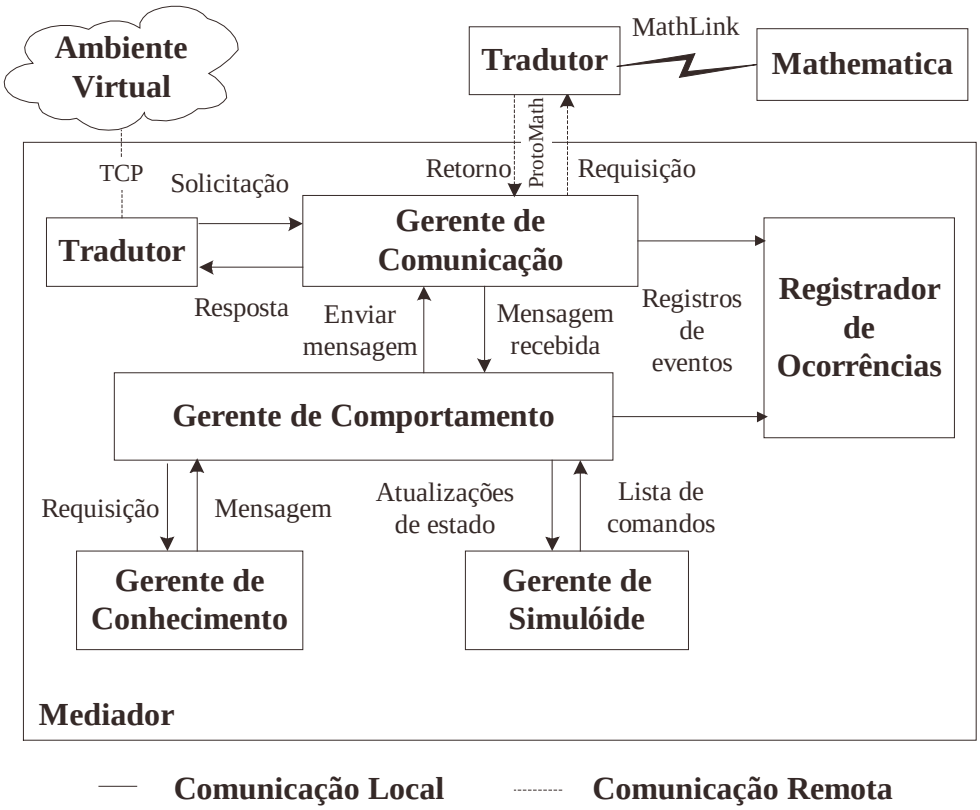


Figura 49. Modelo VEGA adaptado à aplicação Beremiz.

O Beremiz, desenvolvido em Delphi, pode ser usado para diversos fins, tanto como uma ferramenta auxiliar para um professor de Matemática e seus alunos, quanto como uma ferramenta visando o treinamento de pessoas no uso do software Mathematica. Nesse último caso, o Beremiz encarrega-se principalmente de repassar as mensagens para o Mathematica e controlar a fila de requisições ao software.

Considerando-se os requisitos de comunicação do Mathematica e do ambiente virtual ao qual a aplicação mediadora também se conecta, os componentes de comunicação específicos, definidos na aplicação mediadora, possuem as seguintes características.

Tradutores: foram implementados um tradutor para o ambiente virtual que faz uso direto do protocolo TCP e um tradutor para o Mathematica que faz uso da biblioteca MathLink. Ao contrário dos tradutores da aplicação BIA, o tradutor do Mathematica não é de um funcionamento tão simples. Como já foi explicado na seção anterior, ele além de gerenciar a biblioteca MathLink, também realiza transmissões FTP, além de converter arquivos de imagem.

Gerente de comunicação: as unidades de comunicação de cada tradutor, por não terem funcionamento muito complexo, fundiram-se no gerente de comunicação. Portanto, o gerente de comunicação acumulou as funções de gerenciamento dos processos básicos de

conexão com as extremidades, incluindo os processos de validação de acesso a cada sistema, e a tarefa de encaminhar informações relevantes aos processos de conexão e validação de acesso para o registrador de ocorrências, incluindo tentativas de conexão em cada uma das extremidades e respectivos dados (dia, hora, ocorrência e possível causa) sobre sucessos e falhas. Além disso, o gerente de comunicação também informa ao gerente de comportamento se os tradutores estão conectados às suas extremidades.

Um usuário pode fazer requisições ao Beremiz através do envio de mensagens de texto, as quais devem obedecer a um certo formato padrão, de modo que o Beremiz compreenda a que requisição corresponde a mensagem recebida. Para simplificar o reconhecimento dessas requisições, as mesmas foram divididas em três categorias de acordo com o tipo de retorno esperado do Mathematica. A Tabela 5 mostra os tipos das requisições e os seus respectivos retornos.

Tabela 5. Tipos das Requisições ao Beremiz e seus retornos.

Tipo da Requisição	**Retorno**
Defina	Nenhum
Calcule	Expressão em forma de texto
Desenhe	Imagem

Para cada requisição, podem ser definidas uma ou mais especificações dos formatos de sentença correspondentes. O professor, ou qualquer outra pessoa que saiba trabalhar com o Mathematica, que vai utilizar o Beremiz como ferramenta auxiliar é o responsável pela definição desses formatos. Para que isso seja feito de forma personalizada e, conseqüentemente, possa atender vários tipos de usuários, o Beremiz fornece uma interface de configuração, que pode ser vista na Figura 50. Todas as interfaces da aplicação Beremiz estão ilustradas no Anexo 5.

Essa interface permite o cadastro das diversas funções do Mathematica, incluindo: o número e os tipos dos argumentos; a definição do tipo da respectiva requisição; a indicação de se o Beremiz deve enviar o retorno para todos os usuários que estão na sala; e, finalmente, o formato da sentença para a função do Mathematica correspondente. Se o usuário já possuir uma certa familiaridade com a linguagem do Mathematica, ele pode utilizá-la, bastando para tanto informar ao Beremiz qual o tipo daquela requisição. Isso se faz necessário para que o Beremiz saiba qual o tipo do retorno aguardado do Mathematica. Os arquivos com os dados provenientes das interfaces de configuração são ilustrados no Anexo 9.

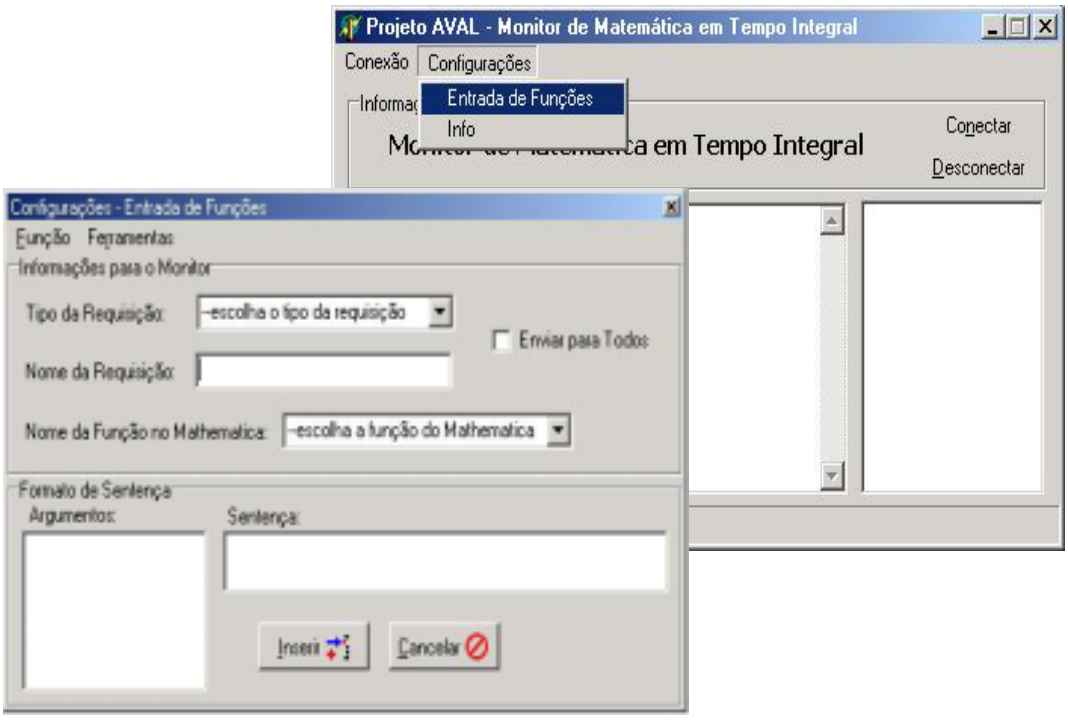


Figura 50. Configuração personalizada dos formatos das sentenças.

Embora a estrutura da aplicação mediadora permita utilizar praticamente qualquer mecanismo para o controle da concorrência das requisições ao Beremiz, encontra-se implementado nessa aplicação, um mecanismo simples que permite atender um usuário de cada vez, inserindo em uma fila de espera as requisições que chegam enquanto o Beremiz está ocupado.

Desse modo, visando atender as requisições dos usuários, os seguintes componentes foram implementados.

Gerente de comportamento: assim como na aplicação BIA, a interação dos demais usuários do ambiente virtual com o Beremiz também origina solicitações que são repassadas ao gerente de comportamento através do tradutor e do gerente de comunicação, explicitados anteriormente. A maior parte das solicitações exige um tratamento mais aprofundado. Nesses casos, o gerente de comportamento repassa as mensagens ao gerente de conhecimento. Mais detalhes sobre o funcionamento do gerente de comportamento encontra-se no diagrama de seqüência do Beremiz no Anexo 7.

Gerente de conhecimento: o gerente de conhecimento uniu-se à unidade de conhecimento específica do Mathematica. O tratamento das mensagens oriundas do ambiente virtual repassadas pelo gerente de comportamento gera requisições no formato utilizado pelo Mathematica, a partir dos dados entrados na interface de configuração. Para exemplificar o

papel do gerente de conhecimento, a Tabela 6 mostra algumas requisições geradas a partir de solicitações dos usuários.

Tabela 6. Solicitações feitas por usuários e suas respectivas requisições geradas para o formato do Mathematica.

Solicitação na linguagem do usuário	Requisição no formato do Mathematica
Desenhe3d Sen(x y) com {x,0,4} e {y,1,6}	Plot3D[Sin[x y], {x,0,4}, {y,1,6}]
MinLocal 3 Sen(x) partindo de x=2	FindMinimum[3 Sin[x], {x, 2}]
Defina f(x) = x + 7	f[x_]=x + 7

Gerente de simulóide: possui as mesmas funcionalidades do gerente de simulóide da aplicação BIA, que são controle de posicionamento, de animação e de expressões faciais específicos do Beremiz. A construção do componente também é igual à da aplicação BIA. O arquivo de script de ações do Beremiz está ilustrado no Anexo 8.

Registrador de ocorrências: semelhante ao registrador de ocorrências da aplicação BIA. Ele armazena em um arquivo de texto informações gerais sobre as conexões mantidas entre o mediador e suas extremidades (ambiente virtual e Mathematica), incluindo data, hora e tipo de ocorrência (conexão realizada com sucesso ou falha), além de registrar as interações dos demais usuários do ambiente virtual com a Beremiz.

O diagrama de classe da aplicação Beremiz, ilustrado no Anexo 6, mostra como cada componente descrito foi implementado.

Um cenário possível de interação entre o Beremiz, o professor (Creto) e uma aluna (Emanuele) é mostrado a seguir:

<Emanuele>: Professor, a função f(x) = 3 Sen(x)+ x possui um mínimo local entre x=2 e x=6?

<Creto>: Que tal se você pedisse ao Beremiz para calcular esse ponto, se ele existir?

<Emanuele>: Como devo digitar a pergunta?

<Creto>: Aguarde um momento porque o Beremiz ainda não consegue compreender esse tipo de pergunta, mas vou ensiná-lo agora...

Nesse momento o professor seleciona o formulário de configurações na interface do Beremiz e acrescenta a função MinLocal correspondente à função do Mathematica FindMinimum, informando à aplicação todos os dados necessários.

<Creto>: Pronto, você pode perguntar assim: Beremiz, MinLocal f partindo de x = 2, mas antes não se esqueça de definir a função f.

<Emanuele>: *Beremiz, defina f(x) = 3 Sen(x)+ x.*

<Beremiz>: *Definido f(x) = 3 Sen(x)+ x.*

<Emanuele>: *Beremiz, MinLocal f(x) partindo de x = 2*

<Beremiz>: *{1.54412, {x -> 4.37255}}*

<Creto>: *Emanuele, você pode saber se existe um mínimo local nesse intervalo de outro modo, pedindo para o Beremiz mostrar o gráfico da função f no intervalo dado. Use a função 'desenhetodos', assim seus colegas também poderão visualizar o gráfico.*

<Emanuele>: *Beremiz, desenhetodos f(x) com x entre 2 e 6.*

<Beremiz>: *Gráfico sendo desenhado! Certifique-se de que sua aceleradora está ligada.*

O momento final desse cenário é ilustrado na Figura 51, onde o professor Creto e a aluna Emanuele podem visualizar o gráfico requisitado ao Beremiz no Quadro Branco da sala de Matemática.

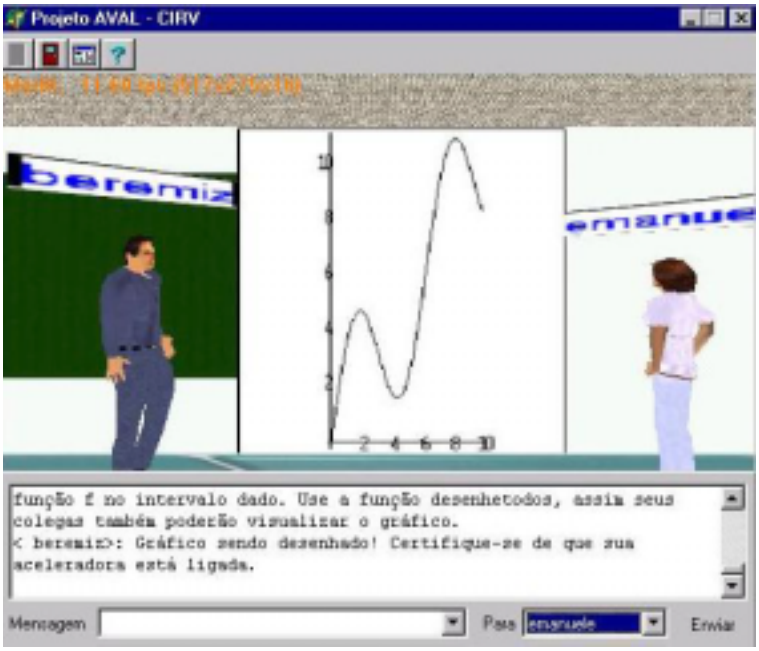


Figura 51. Visão do professor Creto no momento final da interação.

Capítulo VI

Conclusões e Recomendações

Nos últimos anos, a realidade virtual tem sido explorada em um número cada vez maior de aplicações, permitindo a colaboração entre os participantes do ambiente virtual mesmo que eles se encontrem geograficamente distantes uns dos outros. Em paralelo ao desenvolvimento das aplicações de realidade virtual, houve um avanço acelerado do software especializado nas diversas áreas do conhecimento. A integração de aplicações externas a ambientes de realidade virtual mostra-se como uma tarefa bastante complexa, mas que pode tornar disponíveis importantes recursos a todos os usuários desse tipo de sistema. A resolução desse problema constituiu o objetivo principal dessa dissertação.

Para atingir esse objetivo, foi executada a análise de algumas aplicações baseadas em realidade virtual e da importância de utilizar os serviços de outras aplicações a partir de mundos virtuais. Também foi realizado um levantamento dos tipos das abordagens de integração de aplicações externas a ambientes virtuais, citando exemplos de sistemas de realidade virtual que se enquadram nesses tipos. Foi feito, também, um estudo do uso de mediadores para a integração de dados e informações em outras áreas do conhecimento.

A partir de todas as informações obtidas, foi proposto o modelo VEGA – modelo de integração de aplicações externas a ambientes virtuais através de aplicações mediadoras e de simulóides. Para comprovar a viabilidade do modelo foram desenvolvidos dois estudos de casos, a BIA e o Beremiz. A BIA interliga serviços de dicionários eletrônicos a um ambiente virtual em rede para o aprendizado de línguas, e o Beremiz interliga o software Mathematica (um software comercial de computação simbólica) a uma sala virtual de matemática.

6.1 Principais contribuições

A principal contribuição desse trabalho é a definição do modelo VEGA, que integra, a ambientes virtuais, aplicações externas cujos serviços sejam imprescindíveis aos usuários de tais ambientes. Baseada no conceito de aplicações mediadoras e no uso de simulóides, esse tipo de solução possibilita uma maior facilidade nessa integração, pois se

descarta a necessidade de reimplementação das aplicações originais. Utilizando-se do mesmo modelo de integração apresentado nesse trabalho, inúmeras outras aplicações podem tornar-se disponíveis em praticamente qualquer ambiente virtual.

Os dois protótipos desenvolvidos como estudos de casos do modelo também constituem importantes contribuições, pois podem ser utilizados como ferramentas auxiliares em aplicações educacionais.´

A BIA mostrou-se uma ferramenta auxiliar bastante útil no aprendizado de inglês de um curso para guias de turismo.

O Beremiz pode ser usado para diversos fins, tanto como uma ferramenta auxiliar para um professor de Matemática e seus alunos, quanto como uma ferramenta para treinamento de pessoas no uso do software Mathematica. O Beremiz é uma ferramenta versátil que simplifica a forma de interação entre o usuário e o Mathematica, graças à interface interativa de configuração das requisições.

6.2 Recomendações

A reusabilidade dos componentes do modelo VEGA pode ser ampliada através da criação de *frameworks* para a geração automática de componentes, principalmente dos tradutores (porque dependem somente dos tipos de interface e dos protocolos de comunicação). Os demais componentes podem ser implementados da melhor forma a atenderem as especificações das aplicações externas e dos ambientes virtuais.

A capacidade da BIA e do Beremiz de adquirir e distribuir conhecimento pode ser estendida, através da inclusão de um analisador léxico e de um sintático, integrados para o reconhecimento de linguagem natural escrita. O aperfeiçoamento da reação a eventos, por parte dos simulóides, pode ser conseguido através do estabelecimento de regras específicas de comportamento e do emprego de um sistema especialista que gerencie as mesmas. Funções de síntese de voz e *lipsync* (sincronização labial) podem ser incluídas, a fim de tornar os simulóides mais completos e intuitivos.

O Beremiz pode também adquirir a capacidade de, ao invés de retornar uma imagem de um objeto tridimensional (gráficos no espaço 3D, por exemplo), retornar o próprio objeto tridimensional. Esse objeto pode então ser manipulado pelos usuários do ambiente virtual, bastando que o ambiente virtual em questão suporte o formato retornado pelo Mathematica ou que se incorpore um conversor de formatos na aplicação mediadora do Beremiz.

Referências Bibliográficas

Referências Bibliográficas

Ashish, N. & Knoblock, C. A. (1997). *Semi-automatic wrapper generation for internet information sources*. In Proceedings of the Second IFCIS International Conference on Cooperative Information Systems (CoopIS), Charleston, SC. http://citeseer.nj.nec.com/ashish97semiautomatic.html

Batisti, G., Tarouco, L.M.. (1999). *Telepresença com Realidade Virtual para Gerenciamento de Rede*. Anais do Simpósio Brasileiro de Redes de Computadores – SBRC´99, Brasil.

Bécheiraz P. & Thalmann D. (1996). *A Model of Nonverbal Communication and Interpersonal Relationship between Virtual Actors*. In Proceedings Computer Animation ’96, IEEE Computer Society Press, pp. 58-67.

Begault, D.R. (1994). *3-D Sound for Virtual Reality and Multimedia*. Academic Press, Cambridge, MA.

Bird, S. D. (1993). *Towards a taxonomy of multi-agent systems*. International Journal of Man-Machine Studies, 39:689-704.

Bryson, S. & Levit, C. (1992). *The Virtual Wind Tunnel.* IEEE Computer Graphics and Applications, Vol. 2, No. 4.

Burdea,G. & Coiffet,P. (1994). *Virtual RealityTechnology*. John Wiley & Sons, New York, NY.

Campus Virtual da UFPE. (1998). Centro de Informática da Universidade Federal de Pernambuco. URL: http://www.cin.ufpe.br/~if291/galeria/ufpe/cufpe.wrl.

Carey, M. J.; Haas, L. M.; Schwarz, P. M.; Arya, M.; Cody, W. F.; Fagin, R.; Flickner, M.; Luniewski, A. W.; Niblack, W.; Petkovic, D.; Thomas, J.; Williams, J. H. & Wimmers, E. L. (1995). *Towards heterogeneous multimedia information systems: The Garlic approach*. In Proceedings IEEE RIDE-DOM, Taipei, Taiwan.

Cassell, J. (2000a). *More than Just Another Pretty Face: Embodied Conversational Interface Agent.* Communications of the ACM. 43(4), 70–78.

Cassel, J. (2000b). *Elements of Face-to-Face Conversation for Embodied Conversational Agents.* Embodied Conversational Agents, Cassel, J.; Sullivan, J.; Prevost, S., et al., Eds. Cambridge, MA. MIT Press.

Cruz-Neira, C. et al. (1992). *The CAVE Audio Visual Experience Automatic Virtual Environment.* Communication of the ACM, 35(6):64-72,June.

Csordas, T.J. (1997). *Computerized Cadavers: Shades of Being and Representation in Virtual Reality*. Case Western Reserve University, EUA. http://www.focusing.org/compucad.html.

Çapin, T.K.; Pandzic, I.S.; Thalmann, N.M. & Thalmann, D. (1999). *Avatars in Networked Virtual Environments*. John Wiley & Sons, EUA.

Çapin, T.K.; Pandzic, I.S.; Noser, H.; Thalmann, N.M. & Thalmann, D. (1997). *Virtual Human Representation and Communication in VLNET Networked Virtual Environments.* IEEE Computer Graphics and Applications, Special Issue on Multimedia Highways.

Doenges, P.K.; Çapin, T.K.; Lavagetto, F.; Ostermann, J.; Pandzic, I. S. & Petajan, E. D. (1997). *MPEG-4: Audio/Video and Synthetic Graphics/Audio for Mixed Media*. Image Communication Journal, Vol. 5, No. 4, pp. 433-463.

Franchitti, J. C. & King, R. (1993). *Amalgame: atool for creating interoperating persistent, heterogeneous components*. Advanced Database Systems, pp. 313-336.

Frécon, E. & Stenius, M. (1998). *DIVE: A Scaleable network architecture for distributed virtual environments,* Distributed Systems Engineering Journal (special issue on Distributed Virtual Environments), Vol. 5, No. 3, pp. 91-100.

Frery, A.; Groth, B. Jähnichen, S.; Kelner, J.; Dudziak, T. & Teichrieb, V. (1999). *CoNViRA – Conceptual Navigation in Virtual Reality Applications*. In: German–Brazilian and German–Argentinean Workshop on Information Technology, 5. – 2., 1999, Königswinter, Proceedings. Berlim: GMD FIRST.

Frery, A. & Kelner, J. (2001). *Museu Virtual do Homem do Nordeste.* Computação Gráfica. Centro de Informática da Universidade Federal de Pernambuco. http://www.cin.ufpe.br/~if291/galeria/museuhomem/museu-virtual.htm.

Genesereth, M.R. (1995). *Interoperability: Na Agent-Based Framework.* AI Expert, pp. 34-40.

Hancock, D. (1995). *Viewpoint: Virtual Reality in Search of Middle Ground*. IEEE Spectrum, 32(1):68, January.

Jacobson, L. (1991). *Virtual Reality: A Status Report*, AI Expert, pp. 26-33, August.

Johnson, W. L. & Rickel, J. (1998). *Steve: An animated pedagogical agent for procedural training in virtual environments.* SIGART Bulletin 8: 16-21.

Johnson, W. L.; Rickel, J.; Stiles, R. & Munro, A. (1998). *Integrating pedagogical agents into virtual environments*. Presence: Teleoperators and Virtual Environments 7(6): 523-546.

Kent, P. & Stevenson, I. (1999). *'Calculus in Context': A Study of Undergraduate Chemistry Students' Perceptions of Integration*. Paper given at Psychology of Mathematics Education 23.

Kirner, C. (1998). *Projeto Professor Virtual*, http://www.dc.ufscar.br/~grv/pvirtual.htm.

Kirner, C. (2001). *Sistemas de Realidade Virtual*. Faculdade de Informática. Fundação Eurípedes de Marília. http://www.realidadevirtual.com.br/publicacoes/

Knapik, M. & Johnson, J. (1998). *Developing Intelligent Agents for Distributed Systems: Exploring Architecture, Technologies & Applications*. McGraw-Hill, EUA.

Krueger, M.W. (1991). *Artificial Reality II*, Addison-Wesley, Reading, MA.

Leite Jr, A. J. M. (2000). *Ataxia: Uma Arquitetura para a Viabilização de NVE's voltados para a educação a distância através da Internet*. Dissertação de Mestrado, UFC, Brasil.

Lester, J. C.; Stone, B. A. & Stelling, G. D. (1999). *Lifelike pedagogical agents for mixed-initiative problem solving in constructivist learning environments.* User Modeling and User-Adapted Interaction 9:1 44.

Lester, J. C.; Converse, S. A.; Kahler, S. E; Barlow, S. T.; Stone, B. A. & Bhogal, R. (1997a). *The Persona Effect: Affective Impact of Animated Pedagogical Agents*. Proceedings of CHI'97(Human Factors in Computing Systems). pp. 359–366.

Lester, J. C.; Converse, S. A.; Stone, B. A.; Kahler, S. E. & Barlow, S. T. (1997b). *Animated Pedagogical Agents and Problem-Solving Effectiveness: A Large-Scale Empirical Evaluation*. Proceedings of Eighth World Conference on Artificial Intelligence in Education. pp. 23–30.

Lester, J. C.; FitzGerald, P. J. & Stone, B. A. (1997c). *The Pedagogical Design Studio: Exploiting Artifact-Based Task Models for Constructivist Learning*. Proceedings of the Third International Conference on Intelligent User Interfaces. pp. 155–162.

Lester, J. C. & Stone, B. A. (1997). *Increasing Believability in Animated Pedagogical Agents*. Proceedings of the First International Conference on Autonomous Agents. pp. 16–21.

Lester, J. C.; Stone, B. A.; O'Leary, M. A. & Stevenson, R. B. (1996). *Focusing Problem Solving in Design-Centered Learning Environments*. Proceedings of the Third International Conference on Intelligent Tutoring Systems. pp. 475–483.

Macedonia, M. R. (1995). *A network software architecture for large scale virtual environments.* Tese de Doutorado, Naval Postgraduate School. Monterey, CA.

Macedonia, M. R.; Brutzaman, D. P.; Zyda, M. J.; Pratt, D. R.; Braham, P. T.; Falby, J. & Locke, J. (1995). *NPSNET: A Multi-Player 3D Virtual Environment Over the Internet.* Procedures of Symposium on Interactive 3D Graphics, ACM, New York.

Machado, L.S. (1995). Conceitos Básicos da Realidade Virtual. INPE, Brasil.

Maple (2001), http://www.maplesoft.com/products/Maple6/maple6info.html.

Marques, E.; Vidal, C. A.; Leite Jr, A. J. M. & Almendra, C. C. (2001). *Beremiz – Integrando o Mathematica a um Ambiente Virtual em Rede.* Anais do 4th SBC Symposium on Virtual Reality (SVR 2001), Outubro.

Matlab (2001), http://www.mathworks.com/products/matlab.

Miller, D. & Thorpe, J. A. (1995). *SIMNET: The advent of simulator networking*. Proceedings of the IEEE 83(8):1114-1123.

Mine, M. R. (1997). *ISSAC: A Meta-CAD System for Virtual Environments*. Computer-Aided Design.

Mylopoulos, J.; Gal, A.; Kontogiannis, K. & Stanley, M. (1996). *A Generic Integration Architecture for Cooperative Information Systems.* Conference on Cooperative Information Systems, pp. 208-217. http:// citeseer.nj.nec.com/mylopoulos96generic. html.

Oracle Corporation. Oracle Transparent Gateways, 1999. http://www.oracle.com/ gateways/html/transparent.html.

Pandzic, I. S.; Çapin, T. K.; Lee, E.; Thalmann, N. M. & Thalmann, D. (1997). *A flexible architecture for Virtual Humans in Networked Collaborative Virtual Environments.* Proceedings Eurographics 97.

Pandzic, I. S.; Çapin, T. K.; Lee, E.; Thalmann, N. M. & Thalmann, D. (1998). *Autonomous Actors in Networked Collaborative Virtual Environments*. Switzerland: MIRALab.

Papakonstantinou, Y; Garcia-Molina, H.; & Ullman, J. (1996). *MedMaker* : *A Mediation System Based on Declarative Specifications*. Proceedings of the 12th International Conference on Data Engineering. New Orleans, La. http:// citeseer.nj.nec.com/ papakonstantinou95medmaker.html

Papakonstantinou, Y; Garcia-Molina, H.; & Widom, J. (1995). *Object Exchange across Heterogeneous Information Sources*. Procedures of ICDE Conference, 251-260. http://citeseer.nj.nec.com/papakonstantinou95object.html.

Pausch, R.; Snoody, J.; Taylor, R.; Watson, S. & Haseltine, E. (1996). *Disney's Alladin: First Steps Toward Storytelling in Virtual Reality.* Procedures of SIGGRAPH'96.

Pope, A. (1989). *The SIMNET network and protocols.* Technical Report 7102. Cambridge, MA: BBN Systems and Technology, July.

Rickel, J. & Johnson, W. L. (1999). *Animated agents for procedural training in virtual reality: Perception, cognition, and motor control.* Applied Artificial Intelligence in Education. 13:343-382.

Rickel, J. & Johnson, W. L. (1997). *Intelligent tutoring in virtual reality: A preliminary report.* Proceedings of the Eighth World Conference on Artificial Intelligence in Education, 294-301. IOS Press.

Rothbaum, B.; Hodges, L.; Kooper, R. Opdyke, D.; Williford, J. & North, M. (1996). *Effectiveness of computer-generated (virtual reality) graded exposure in the treatment of acrophobia.* American Journal of Psychiatry, Vol. 152, No. 4.

Singh, G.; Serra, L.; Prg, W., et al. (1994). *BrickNet: A software toolkit for networked-based virtual environments.* PRESENCE: Teleoperators and Virtual Environments 3(1):19-34.

Singh, G.; Serra, L.; Prg, W., et al. (1995). *BrickNet: Sharing object behaviors on the net.* Proceedings of the Virtual Reality Annual International Symposium (VRAIS'95), 19-25. Los Alamitos, CA: IEEE Computer Society Press.

Singhal, S. & Zyda, M. (1999). *Networked Virtual Environments*. Addison Wesley, EUA.

Slater, M. & Usoh, M. (1994). *Body Centered Interaction in Immersive Virtual Environments.* Artificial Life and Virtual Reality. John Wiley, Chichester.

Souder, T. & Mancoridis, S. (1999). *A Tool for Securely Integrating Legacy Systems into a Distributed Environment*. In Working Conference on Reverse Engineering (WCRE), Atlanta, GA. pp. 47-55. http:// citeseer.nj.nec.com/souder99tool.html.

Stallings, W. (1997). *High-Speed Networks: TCP/IP and ATM Design Principles.* Prentice Hall, EUA.

Stiles, R.; McCarthy, L. & Pontecorvo, M. (1995). *Training Studio: A virtual environment for training.* Workshop on Simulation and Interaction in Virtual Environments (SIVE-95). Iowa City, IW. ACM Press.

Stone, B. A. & Lester, J. C.; (1996), *Dynamically Sequencing an Animated Pedagogical Agent*. Proceedings of the Thirteenth National Conference on Artificial Intelligence. pp. 424–431.

Stytz, M.R. (1996). *Distributed Virtual Environments*. IEEE Computer Graphics and Applications, Vol. 16, No. 33, pp. 19-31.

Teichrieb, V. (1999). *Avatares como Guias Interativos para Auxílio na Navegação em Ambientes Virtuais Tridimensionais.* Dissertação de Mestrado, UFPe, Brasil.

Vidal, C.A. (1999). *Projeto AVAL – Ambientes Virtuais para o Aprendizado de Línguas,* http://www.lcg.dc.ufc.br/aval.

Vidal, C. A.; Leite Júnior, A. J. M.; Almendra, C. C.; Santos, E. M.& Oliveira, J. P. C. (2000). *Uma proposta de Integração de Ferramentas Externas a Ambientes Virtuais através de Simulóides e de Aplicações Mediadoras*. Anais do 3rd Brazilian Workshop on Virtual reality, Brasil.

Wazlawick, R. S.; Fagundes, L. C. & Kirner, T. G. (1999). *Museu Virtual – Ferramenta de Autoria para Criação de Museus de Realidade Virtual para Apoio à Aprendizagem Colaborativa via Internet.* Projeto ProTeM PTI/PEDU, Brasil.

Wiederhold, G. (1992). *Mediators in the architecture of future information systems*. IEEE Computer, 25:38-49. http://citeseer.nj.nec.com/wiederhold92mediators.html.

Wolfram, S. (1996). *The Mathematica Book*, third edition. Cambridge University Press, EUA.

Wolfram (2001). http://www.wolfram.com/solutions/highered/how/campuses.html.

Zyda, M. & Sheehan, J. (Eds.) (1997). *Modeling and Simulation: Linking Entertainment and Defense*. National Academy Press.

Anexos

Anexo 1

Projeto de Interface da Aplicação BIA

Tela de conexão ao ambiente virtual e ao glossário.

Tela principal

Tela de informações

Anexo 2

Diagrama de Classes da Aplicação BIA

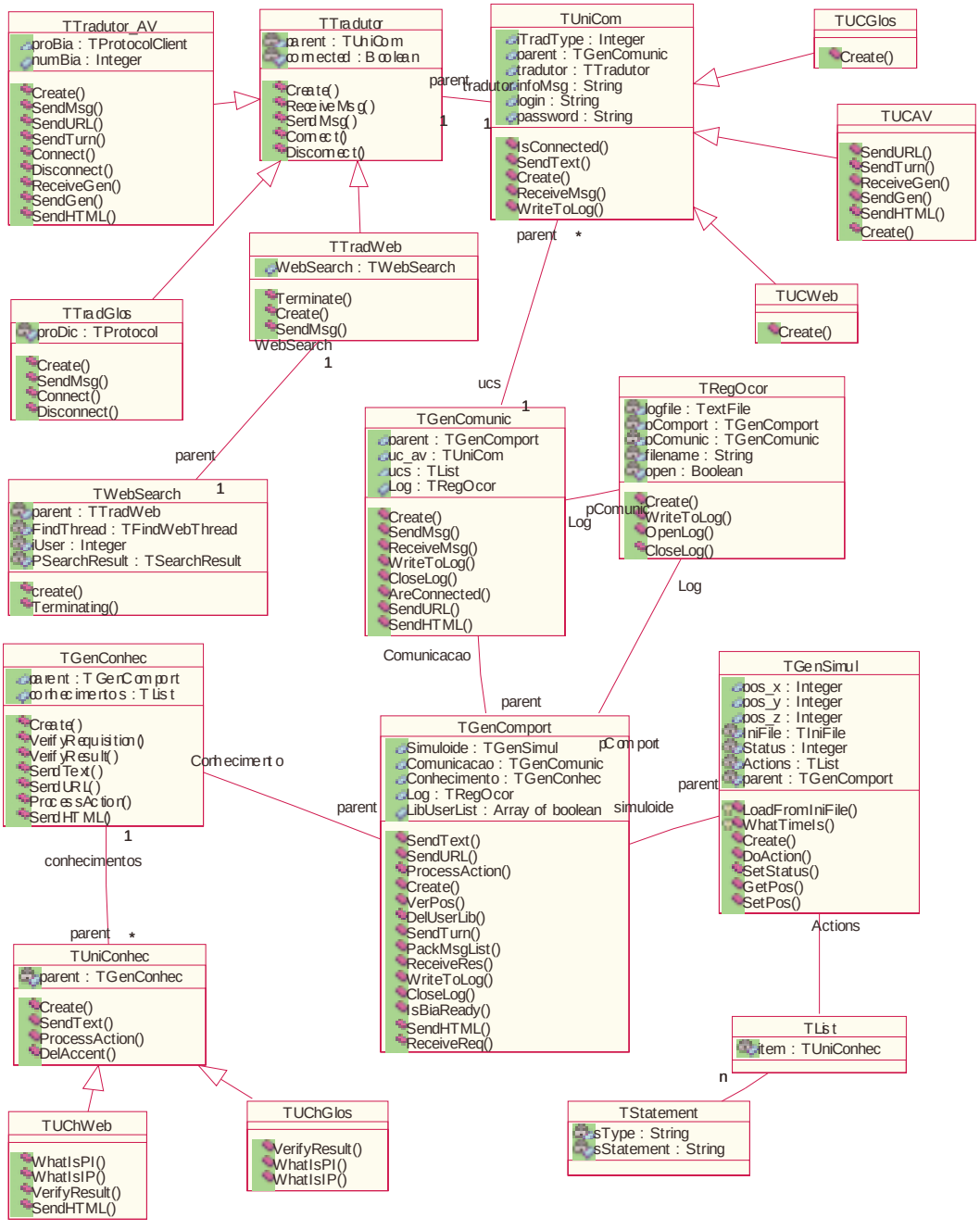


Diagrama de classe da aplicação mediadora da BIA.

Classe: **TTradutor_AV**
Descrição: Tradutor do Ambiente Virtual.
Superclasses: TTradutor
Atributos:
proBia : TprotocolClient; numBia : Integer
Métodos:
Create
Return Class: TTradutor_AV
Arguments: pUC: TUniCom
Documentation: Construtor
SendMsg
Arguments: iDestiny: Integer; Msg: String
Documentation: Envia mensagem de texto para o usuário identificado por iDestiny no ambiente virtual.
SendURL
Arguments: User: Integer; Urls: TStringList
Documentation: Envia links com suas respectivas descrições para o usuário identificado por User do AV.
SendTurn
Arguments: angle: Integer
Documentation: Envia mensagem de rotação em graus.
Connect
Documentation: Mensagem de conexão com o ambiente virtual.
Disconnect
Documentation: Mensagem de desconexão do ambiente virtual.
ReceiveGen
Arguments: user: Integer; Msg: String
Documentation: Recebe mensagem do tipo genérica do ambiente virtual.
SendGen
Arguments: user: Integer; Msg: String
Documentation: Envia mensagem do tipo genérica para o ambiente virtual.
SendHTML
Arguments: iUser: Integer; htmlFile: String
Documentation: Mensagem que contém o arquivo html com o retorno da ferramenta externa web.

Classe: **TTradGlos**
Descrição: Tradutor do Glossário
Superclasses: TTradutor
Atributos:
proDic : TProtocol
Métodos:
Create
Return Class: TTradGlos
Arguments: pUC: TTradWeb
Documentation: Construtor
SendMsg
Arguments: Msg: String; iDestiny: Integer
Documentation: Procedimento chamado ao enviar mensagem para a FE.
Connect

Disconnect

Classe: **TStatement**
Descrição: Armazena a estrutura de cada comando da lista de ações da BIA.
Atributos:
sType : String; sStatement : String
Métodos:
Create
Return Class: TStatement
Arguments: pType: String; pStatement: String

Classe:**TUCAV**
Descrição: Unidade de Comunicação do Ambiente virtual.
Superclasses: TUniCom
Métodos:
SendURL
Arguments: User: Integer; Urls: TStringList
SendTurn
Arguments: angle: Integer
ReceiveGen
Arguments: User: Integer; Msg: String
SendGen
Arguments: User: Integer; Msg: String
SendHTML
Arguments: iUser: Integer; htmlFile: String
Create
Return Class: TUCAV
Arguments: parent: TgenComunic; iType: Integer

Classe: **TUCGlos**
Descrição: Unidade de Comunicação do Glossário.
Superclasses: TUniCom
Métodos:
Create
Return Class: TUCGlos
Arguments: parent: TgenComunic; iType: Integer

Classe: **TUCWeb**
Descrição: Unidade de Comunicação do dicionário da web.
Superclasses: TUniCom
Métodos:
Create
Return Class: TUCWeb
Arguments: parent: TgenComunic; iType: Integer

Classe: **TUChWeb**
Descrição: Unidade de Conhecimento do Babylon.
Superclasses: TUniConhec
Métodos:
WhatIsPI

Return Class: String
Arguments: iUser: Integer; Msg: String
Documentation: Monta a msg do tipo consulta definição Português -> Inglês

WhatIsIP
Return Class: String
Arguments: iUser: Integer; Msg: String
Documentation: Monta a msg do tipo consulta definição Inglês -> Português

VerifyResult
Arguments: SenderNum: Integer; Msg: String
Documentation: Verifica e processa o retorno do dicionário da web.

SendHTML
Arguments: iUser: Integer; htmlFile: String
Documentation: Envia arquivo html contendo retorno do babylon.

Classe: **TUChGlos**
Descrição: Unidade de Conhecimento do Glossário.
Superclasses: TUniConhec
Métodos:

VerifyResult
Arguments: SenderNum: Integer; Msg: String

WhatIsPI
Return Class: String
Arguments: Msg: String

WhatIsIP
Return Class: String
Arguments: Msg: String

Classe: **TUniConhec**
Descrição: Classe genérica da Unidade de Conhecimento
Atributos:

parent : TGenConhec

Métodos:

Create
Return Class: TUniConhec
Arguments: parent: TGenConhec

SendText
Arguments: iDestiny: Integer; iUser: Integer; Msg: String

ProcessAction
Arguments: Action: String; iUser: Integer

DelAccent
Return Class: String
Arguments: sword: String
Documentation: Formata palavras, retirando os acentos.

Generalization:
Client: TUChWeb
Client: TUChGlos

Classe: **TGenSimul**
Descrição: Gerente de Simulóide.
Atributos:

pos_x : Integer; pos_y : Integer; pos_z : Integer; IniFile : TiniFile; Status : Integer; Actions : TList; parent : TGenComport

Métodos:

Create
Return Class: TGenSimul
Arguments: pParent: TGenComport

DoAction
Return Class: TList
Arguments: action: String
Documentation: Dada a ação action, retorna uma lista de comandos correspondentes.

SetStatus
Arguments: iState: Integer
Documentation: Configura o estado da BIA.

GetPos
Return Class: TPoint
Documentation: Retorna as coordenadas x e y da Bia no Ambiente Virtual.

SetPos
Arguments: posX: Integer; posY: Integer; posZ: Integer
Documentation: Configura as posições x, y e z da BIA no Ambiente Virtual.

LoadFromIniFile
Return Class: TList
Arguments: IniFile: TIniFile
Documentation: Carrega a Lista de Ações do arquivo.

WhatTimeIs
Return Class: String
Documentation: Função que retorna a parte do dia do Ambiente Virtual: M: Manhã; T:Tarde e N: Noite.

Classe: **TGenConhec**
Descrição: Gerente de Conhecimento.
Atributos:

parent : TGenComport; conhecimentos : TList

Métodos:

Create
Return Class: TGenConhec
Arguments: pparent: TGenComport
Documentation: construtor.

VerifyRequisition
Return Class: Integer
Arguments: Msg: String; SenderNum: Integer
Documentation: Verifica o tipo de requisição.

VerifyResult
Return Class: Integer
Arguments: iType: Integer; SenderNum: Integer; Msg: String
Documentation: Verifica e processa o tipo de retorno da FE

SendText
Arguments: iDestiny: Integer; iUser: Integer; Msg: String

SendURL
Arguments: iUser: Integer; Urls: TStringList

ProcessAction
Arguments: Action: String; iUser: Integer
SendHTML
Arguments: iUser: Integer; htmlFile: String

Classe: **TTradutor**
Descrição: Classe genérica do Tradutor
Atributos:
parent : TuniCom; connected : Boolean
Métodos:
Create
Return Class: TTradutor
Arguments: pUC: TUniCom
ReceiveMsg
Arguments: SenderNum: Integer; Msg: String
Documentation: Mensagem recebida pelo tradutor (msg) enviada pelo usuário identificado por SenderNum.
SendMsg
Arguments: iDestiny: Integer; Msg: String
Documentation: Envia mensagem (msg) para o usuário identificado por iDestiny.
Connect
Disconnect
Generalization:
Client: TTradutor_AV
Client: TTradGlos

Classe: **TRegOcor**
Descrição: Registrador de Ocorrências
Atributos:
logfile : TextFile; pComport : TgenComport; pComunic : TgenComunic; filename : String; open : Boolean
Métodos:
Create
Return Class: TRegOcor
Arguments: pCp: TgenComport; pCm: TgenComunic; filename: String
WriteToLog
Arguments: Text: String
Documentation: Escreve linha no arquivo de log.
OpenLog
Return Class: boolean
Documentation: Abre o arquivo de log.
CloseLog
Documentation: Fecha o arquivo de log.
Classe: **TGenComport**
Descrição: Gerente de Comportamento
Atributos:
Simuloide : TgenSimul; Comunicacao : TgenComunic;Conhecimento : TGenConhec; Log : TregOcor; LibUserList : Array of boolean
Métodos:

SendText
Arguments: iDestiny: Integer; iUser: Integer; Msg: String
Documentation: Envia mensagens para o Gerente de Comunicação, que as envia para o usuário iUser da extremidade identificada por iDestiny.
SendURL
Arguments: iUser: Integer; Urls: TStringList
Documentation: Envia endereços internet para o usuário identificado por iUser do ambiente virtual.
ProcessAction
Arguments: Action: String; iUser: Integer
Documentation: Processa as ações recebidas do controlador de simuloide enviando os respectivos tipos de mensagem para o ambiente virtual.
Create
Return Class: TGenComport
VerPos
Arguments: SenderNum: Integer; PosX: Integer; PosY: Integer
Documentation: Verifica se o avatar está próximo da BIA.
DelUserLib
Arguments: iUser: Integer
Documentation: Deleta usuário identificado por iUser da lista de usuários que estão na biblioteca.
SendTurn
Arguments: angle: Integer
Documentation: Envia mensagem de rotação para o ambiente virtual.
PackMsgList
Arguments: iSender: Integer; iDestiny: Integer; UserNum: Integer; List: TStringList
ReceiveRes
Arguments: iType: Integer; SenderNum: Integer; Msg: String
WriteToLog
Arguments: Msg: String
CloseLog
IsBiaReady
Return Class: Boolean
SendHTML
Arguments: iUser: Integer; htmlFile: String
ReceiveReq
Arguments: SenderNum: Integer; Msg: String
Documentation: Mensagem recebida do ambiente virtual

Classe: **TUniCom**
Descrição: Classe genérica da Unidade de Comunicação
Atributos:
iTradType : Integer; parent : TgenComunic; tradutor : Ttradutor; infoMsg : String; login : String; password : String
Métodos:
IsConnected
Return Class: Boolean
Documentation: Função que retorna se o tradutor está conectado a aplicação externa ou não.

SendText
Arguments: iDestiny: Integer; Msg: String
Create
Return Class: TUniCom
Arguments: parent: TgenComunic; iType: Integer
ReceiveMsg
Arguments: SenderNum: Integer; Msg: String
WriteToLog
Arguments: Msg: String
Generalization:
Client: TUCAV
Client: TUCGlos
Client: TUCWeb

Classe: **TGenComunic**
Descrição: Gerente de Comunicação.
Atributos:
parent : TGenComport; uc_av : TuniCom; ucs : Tlist; Log : TRegOcor
Métodos:
Create
Return Class: TGenComunic
Arguments: pParent: TGenComport
SendMsg
Arguments: iCom: Integer; iDestiny: Integer; Msg: String
Documentation: Envia mensagens de texto para a Unidade de Comunicação especificada por iCom.
ReceiveMsg
Arguments: iType: Integer; SenderNum: Integer; Msg: String
Documentation: Procedimento chamado ao chegar mensagens para a Bia.
WriteToLog
Arguments: Msg: String
Documentation: Escreve evento no registrador de ocorrências.
CloseLog
AreConnected
Return Class: Boolean
SendURL
Arguments: iUser: Integer; Urls: TStringList
Documentation: Envia endereços Internet com suas respectivas descrições para o usuário identificado por iUser doAmbiente Virtual.
SendHTML
Arguments: iUser: Integer; htmlFile: String

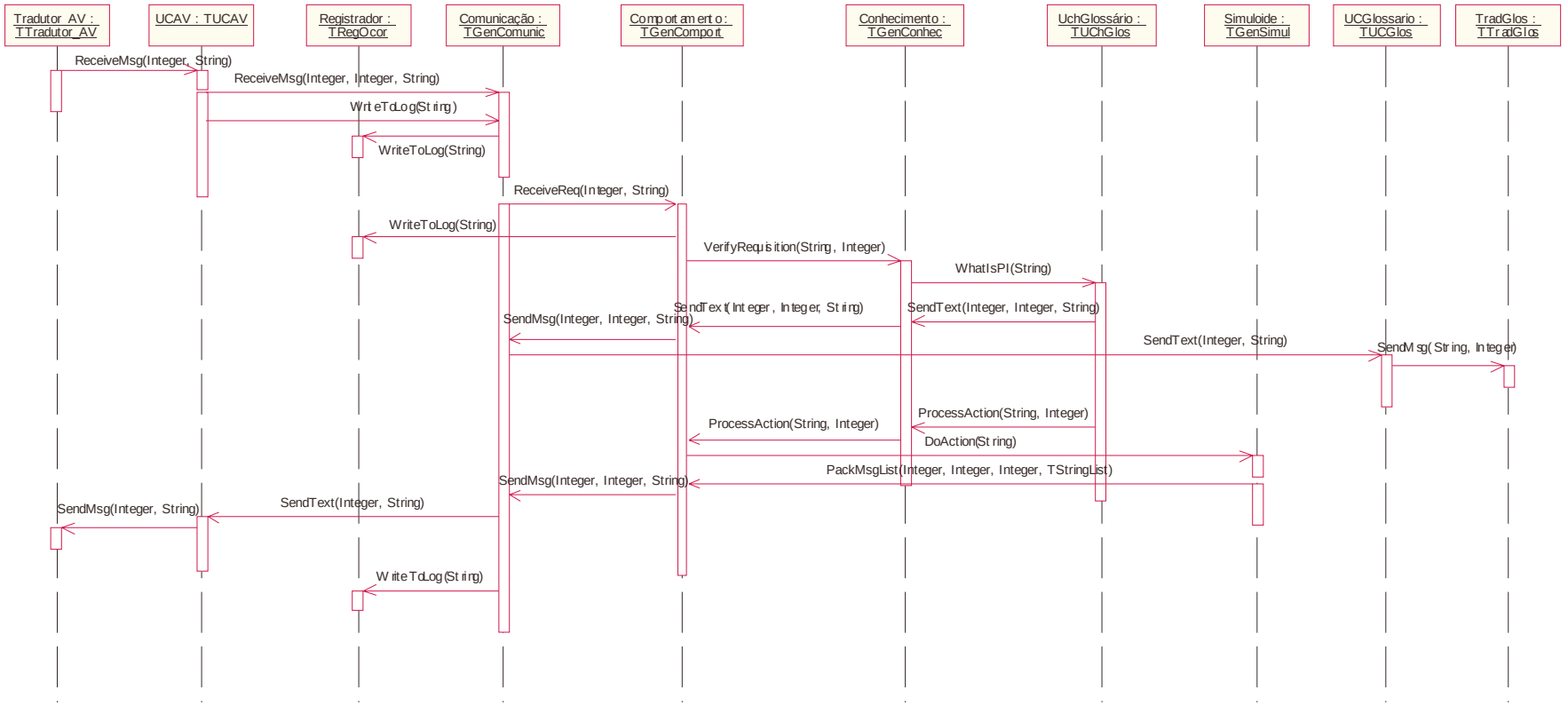
Classe: **TTradWeb**
Descrição: Tradutor do dicionário da web.
Superclasses: TTradutor
Atributos:
WebSearch : TWebSearch
Métodos:
Terminate
Arguments: str: String

Create
Return Class: TTradWeb
Arguments: pUC: TUniCom
SendMsg
Arguments: iDestiny: Integer; Msg: String

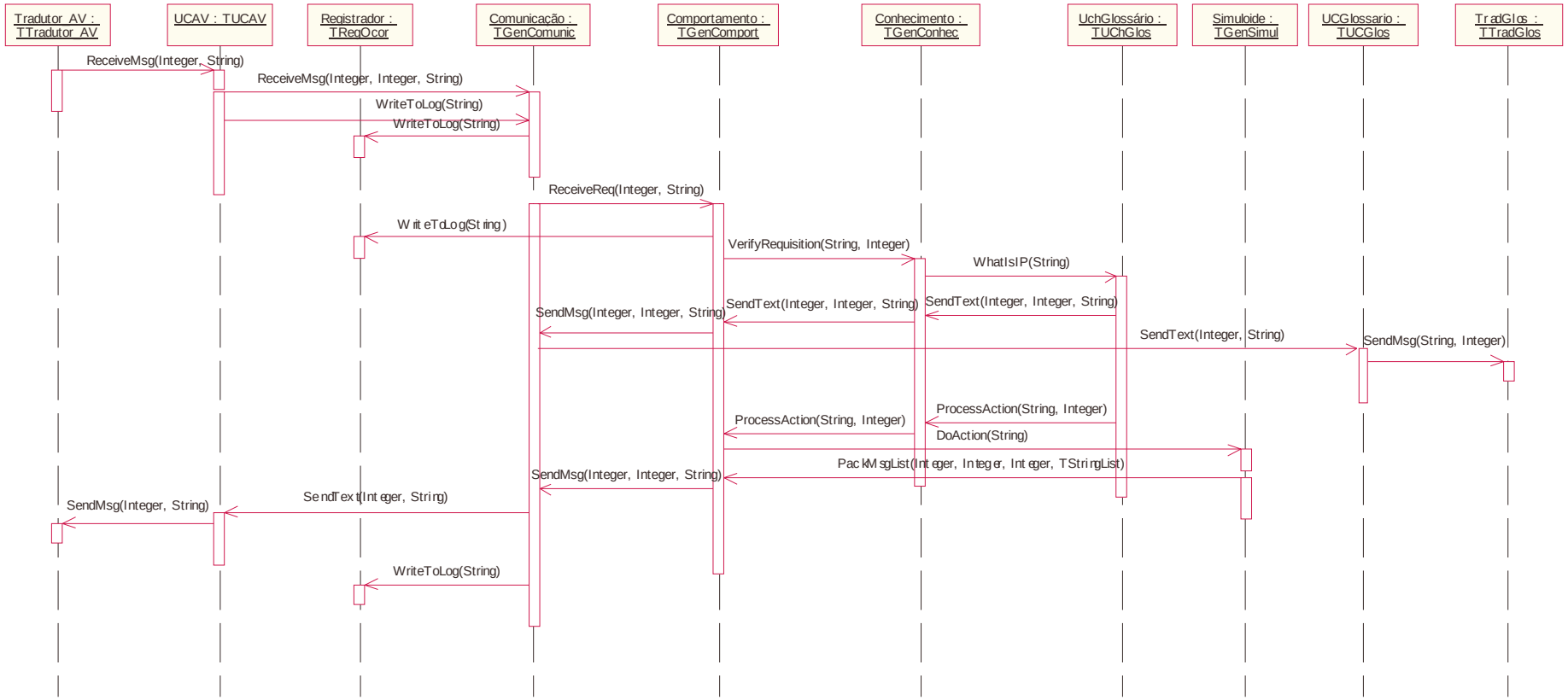
Classe: **TWebSearch**
Descrição: Classe de busca na web
Atributos:
parent : TtradWeb; FindThread : TfindWebThread; iUser : Integer; PSearchResult : TSearchResult
Métodos:
Create
Return Class: TWebSearch
Arguments: parent: TtradWeb; strurl: String; iUser: Integer
Terminating
Arguments: str: String

Anexo 3

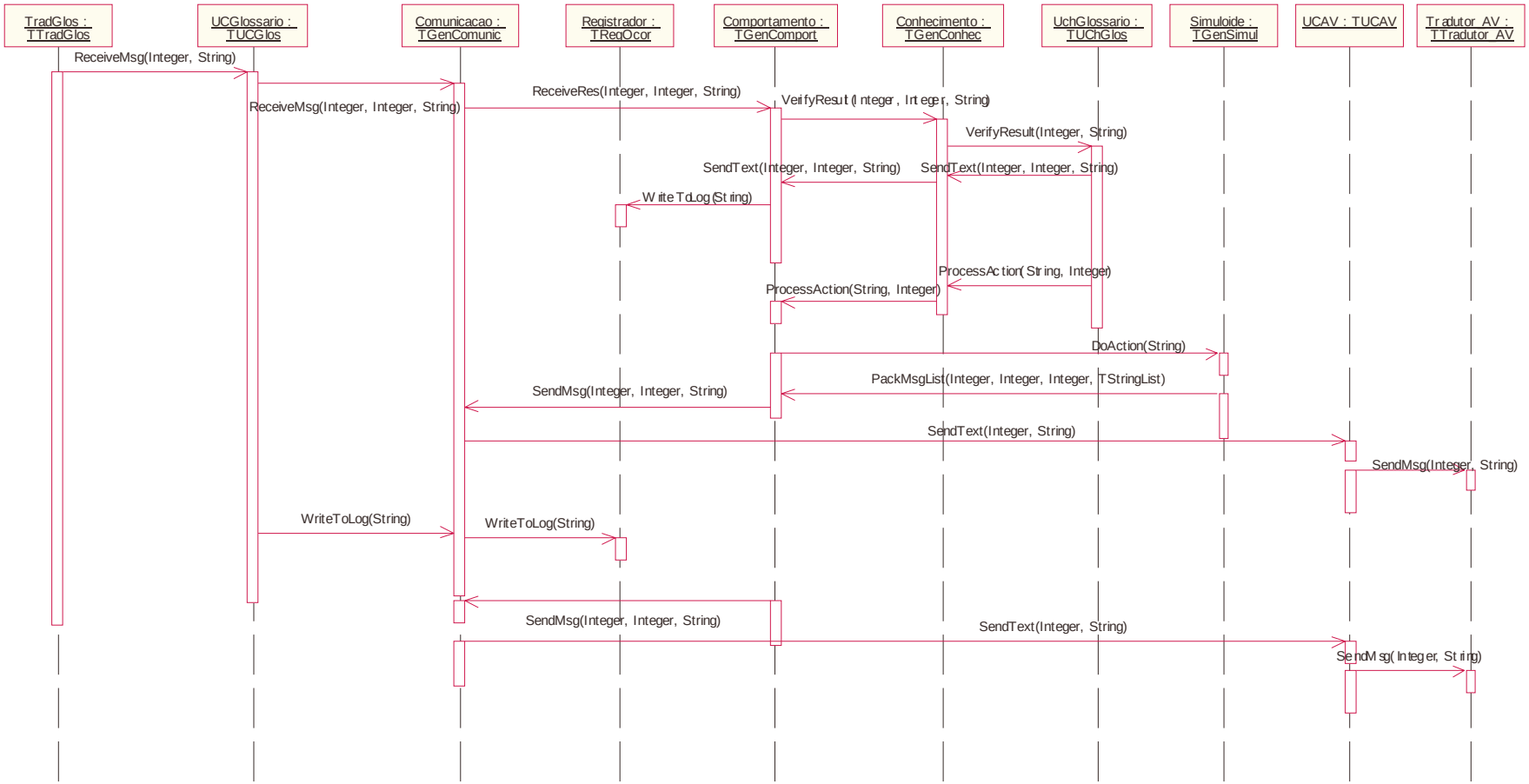
Diagrama de Seqüência da Aplicação BIA



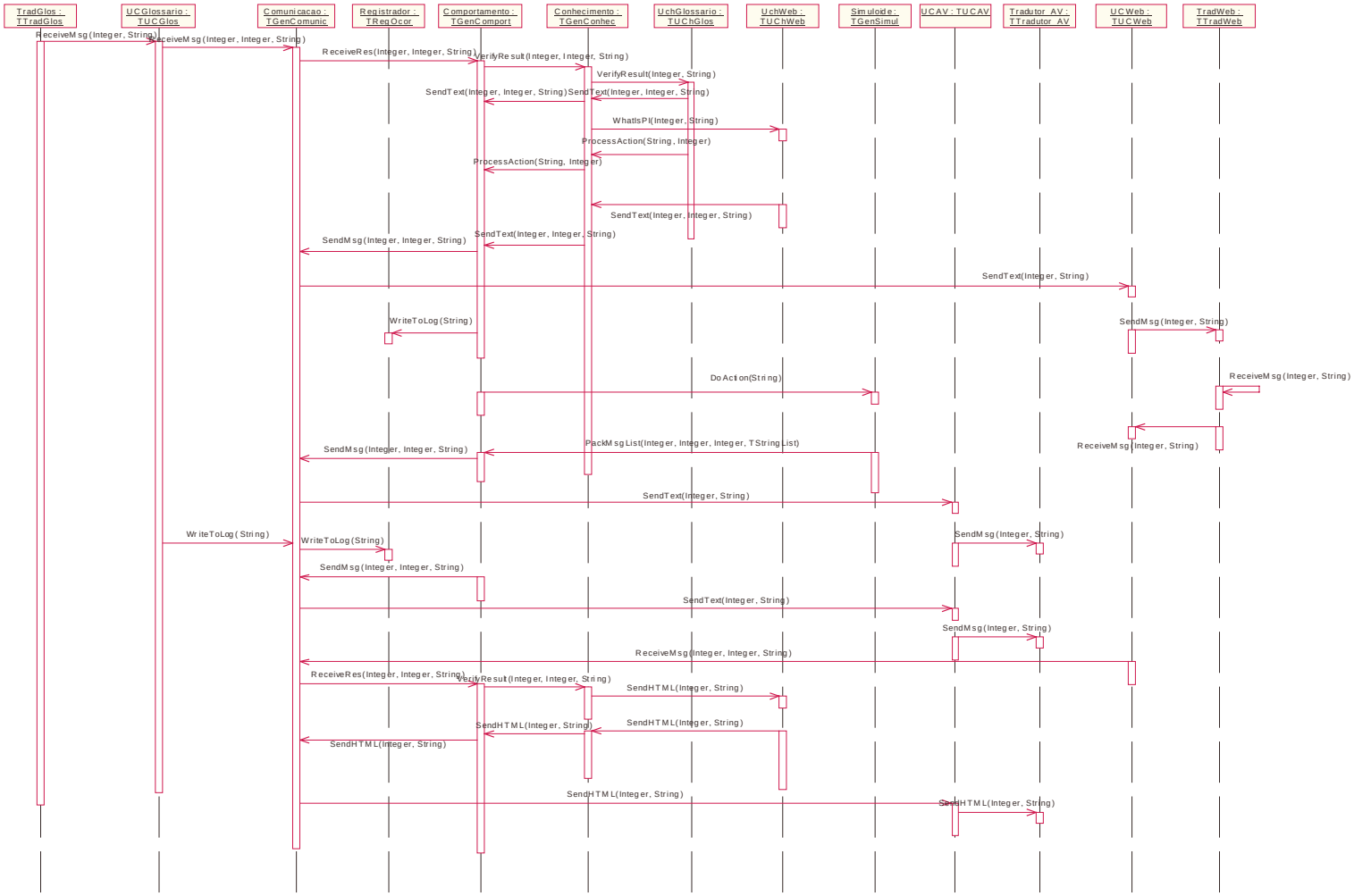
Cenário: Trajeto da requisição de tradução Português-Inglês do ambiente virtual até o glossário.



Cenário: Trajeto da requisição de tradução Inglês-Português do ambiente virtual até o glossário.



Cenário: Retorno positivo do glossário para requisição de tradução.



Cenário: Retorno negativo do glossário para requisição de tradução e requisição ao babylon.

Anexo 4

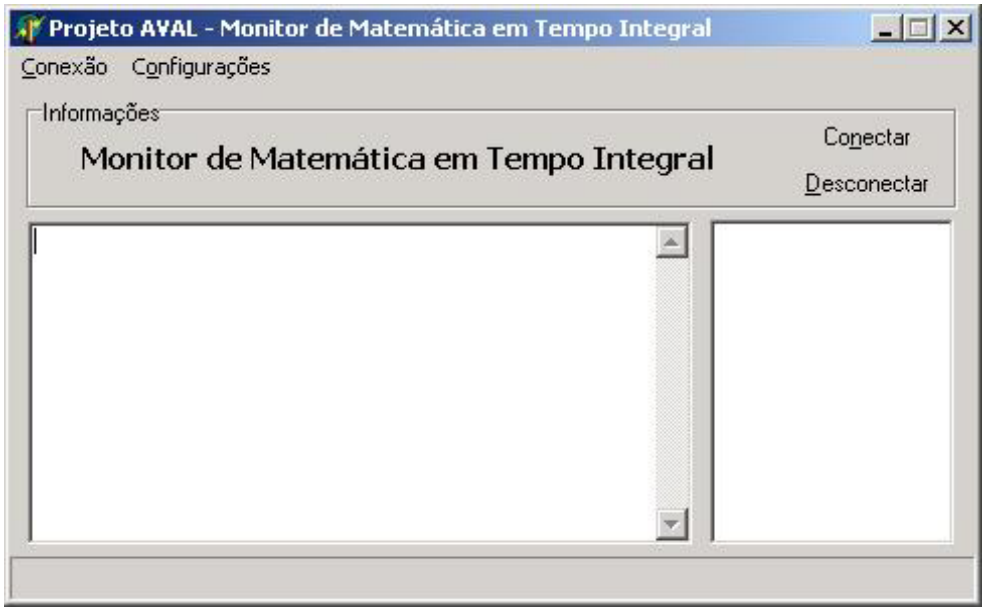
Arquivo de Script de Ações do Simulóide Controlado pela Aplicação BIA

```
;; Arquivo de inicialização do controle de simulóide da BIA
;; Os comandos devem ser colocados na ordem em que eles serão executados
;; Posteriormente colocar o tipo de mensagem a que eles correspondem
;; O primeiro caractere depois do = definirá o tipo:
;; T: mensagem de texto
;; M: movimentação do simulóide (ação)
;; S: som
;; E: expressão facial
;; D: deslocamento

;Saudação Inicial Manhã
[SIM]
0=D,Virar para usuário
1=E,sorrindo
2=T,Good Morning!
3=S,bomdia.mp3
4=T,May I help you?
5=E,normal

;Saudação Inicial Tarde
[SIT]
0=D,Virar para usuário
1=E,sorrindo
2=T,Good Afternoon!
3=S,boatarde.mp3
4=T,May I help you?
5=E,normal

;Saudação Inicial Noite
[SIN]
0=D,Virar para usuário
1=E,sorrindo
2=T,Good Evening!
3=S,boanoite.mp3
4=T,May I help you?
5=E,normal

;Requisição de usuário
[REQ]
0=E,sorriso enigmático
1=T,Please wait a moment...

;Resposta de Tradução com Sucesso
[RESPDEFSUC]
0=E,sorrindo
1=S,acheidesc.mp3

;Resposta de Tradução com Fracasso
[RESPDEFFRA]
0=E,decepcionado
1=S,nacheidesc.mp3

;Requisição não compreendida
[REQINCOMP]
0=E,desconsertado
1=S,perdao.mp3
2=T,I am sorry I do not understand your question.

;Posição Normal
[NORMALPOS]
0=E,normal
```

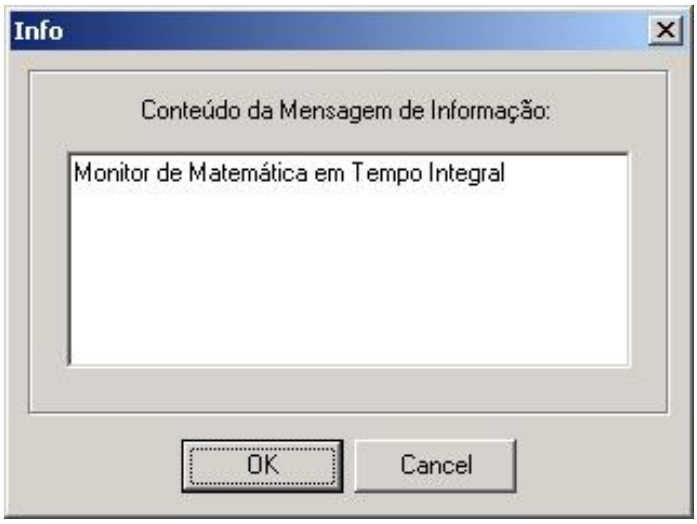
Anexo 5

Projeto de Interface da Aplicação Beremiz

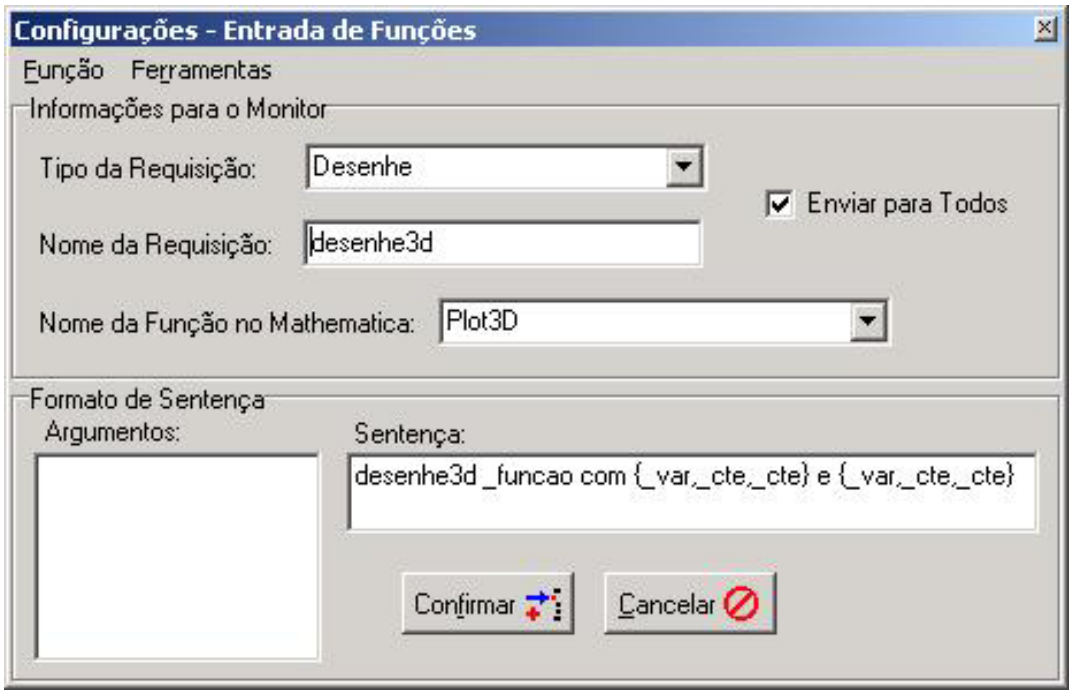
Tela de conexão ao ambiente virtual e ao tradutor do Mathematica.



Tela principal.

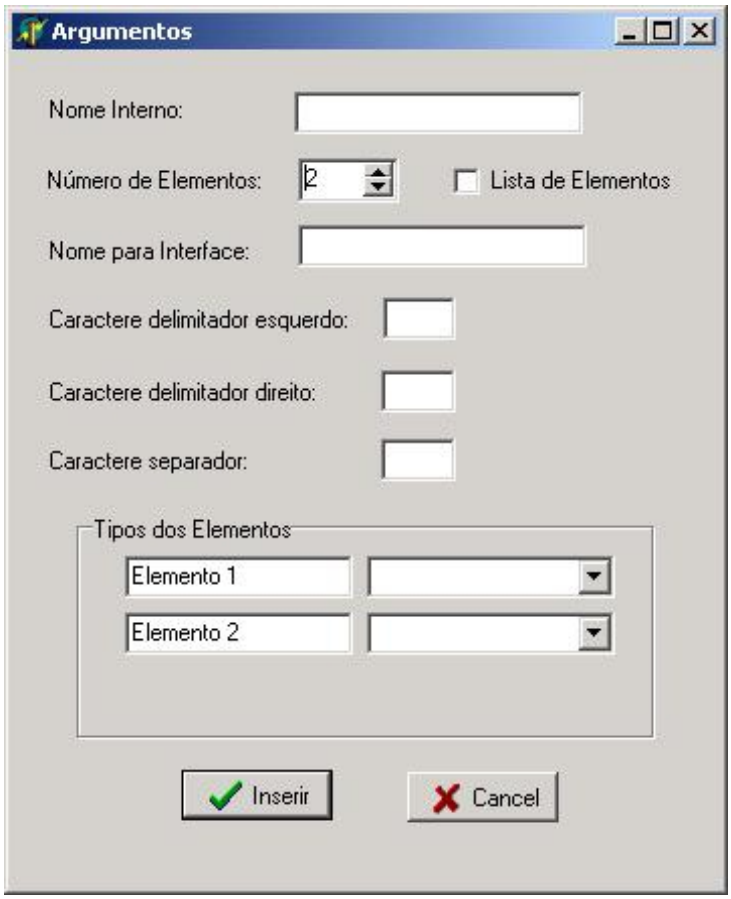


Tela de informações.

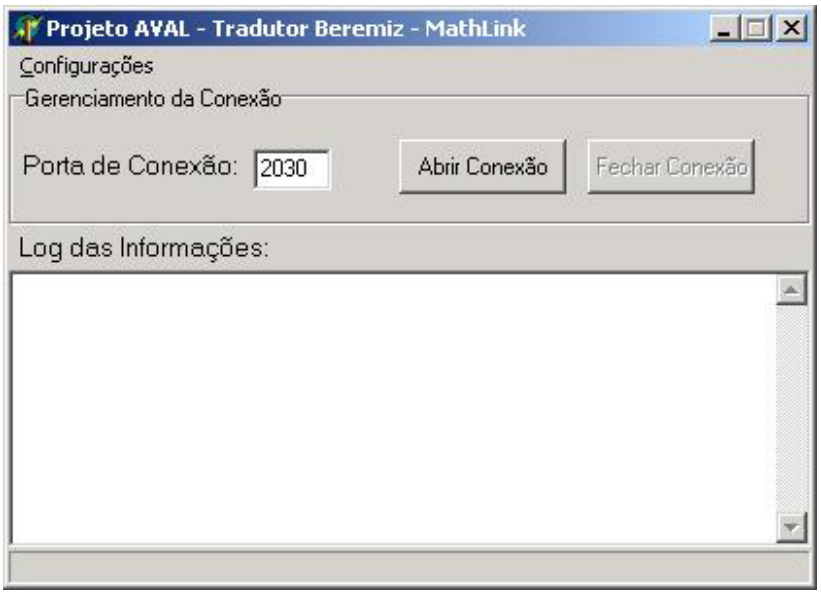


Tela de dados das funções utilizadas na linguagem do usuário.

Tela de dados das funções do Mathematica.



Tela de dados dos tipos de argumentos da s funções do Mathematica.



Tela principal do tradutor do Mathematica.

Tela de configurações de FTP do tradutor.

Anexo 6

Diagrama de Classes da Aplicação Beremiz

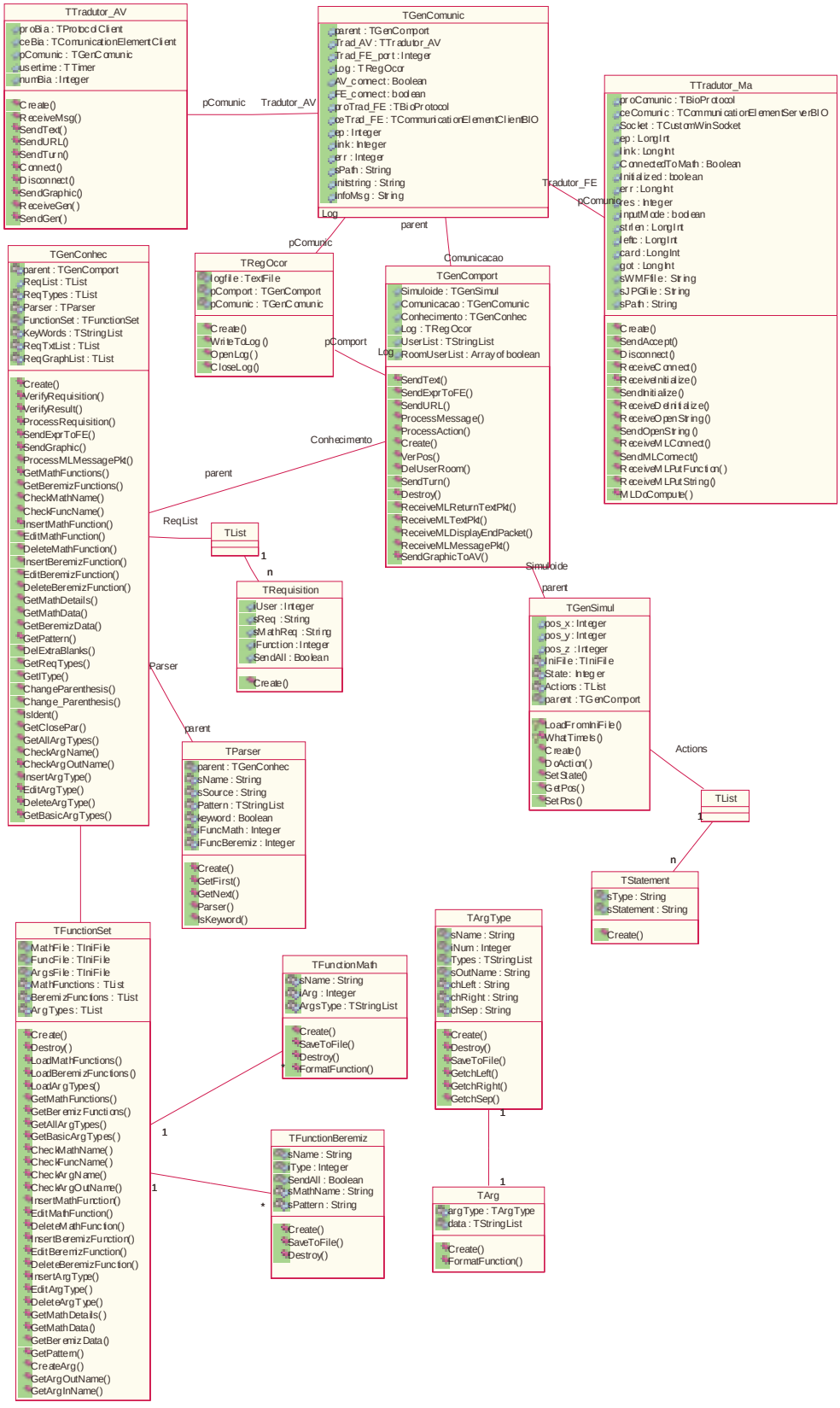


Diagrama de classe da aplicação mediadora do Beremiz.

Classe: **TStatement**
Descrição: Armazena a estrutura de cada comando da lista de ações do Beremiz.
Atributos:
sType : String; sStatement : String
Métodos:
Create
Return Class: TStatement
Arguments: pType: String; pStatement: String

Classe: **TTradutor_AV**
Descrição: Tradutor do Ambiente Virtual.
Atributos:
proBia : TprotocolClient; ceBia : TcomunicationElementClient; pComunic: TgenComunic; usertime : Ttimer; numBia : Integer
Métodos:
Create
Return Class: TTradutor_AV
Arguments: pCC: TGenComunic
Documentation: Construtor
ReceiveMsg
Arguments: SenderNum: Integer; Msg: String
Documentation: Método chamado ao chegar mensagem no tradutor do AV. O mesmo repassa Msg e o número do seu remetente (SenderNum) para o Gerente de Comunicação
SendText
Arguments: User: Integer; Msg: String
Documentation: Envia mensagem de texto para o usuário identificado por User no ambiente virtual.
SendURL
Arguments: User: Integer; Urls: TStringList
Documentation: Envia links com suas respectivas descrições para o usuário identificado por User do AV.
SendTurn
Arguments: angle: Integer
Documentation: Envia mensagem de rotação em graus.
Connect
Documentation: Método chamado ao ser estabelecida a conexão com o ambiente virtual.
Disconnect
Documentation: Método chamado quando o Beremiz é desconectado do Ambiente Virtual.
SendGraphic
Arguments: User: Integer; Msg: String
Documentation: Envia sinalização que o arquivo de imagem está disponível no servidor de ftp.
ReceiveGen
Arguments: User: Integer; Msg: String
SendGen
Arguments: User: Integer; Msg: String

Classe: **TTradutor_Ma**
Descrição: Tradutor da Ferramenta Externa
Atributos:

proComunic : TbioProtocol; ceComunic : TCommunicationElementServerBIO; Socket : TcustomWinSocket; ep : LongInt; link : LongInt; ConnectedToMath : Boolean; Initialized : Boolean; err : LongInt; res : Integer; inputMode : Boolean; strlen : LongInt; leftc : LongInt; card : LongInt; got : LongInt; sWMFfile : String; sJPGfile : String; sPath : String

Métodos:

- Create
  - Arguments: pCC: TGenComunic
  - Documentation: Construtor
- SendAccept
  - Arguments: Login: String; Result: char
  - Documentation: Envia mensagem para o BIO de aceitação de conexão
- Disconnect
  - Documentation: Método chamado quando a BIA é desconectada da FE.
- ReceiveConnect
  - Arguments: Socket: TcustomWinSocket; Login: String; Password: String
  - Documentation: Recebe mensagem de conexão do Bio
- ReceiveInitialize
  - Arguments: p: LongInt
  - Documentation: Recebe Pedido de Inicialização da Biblioteca do Mathematica
- SendInitialize
  - Arguments: result: LongInt
  - Documentation: Envia Resposta de Inicialização da Biblioteca para o Controlador de Comunicação
- ReceiveDeInitialize
  - Arguments: env: LongInt
  - Documentation: Recebe pedido de Desligamento da Biblioteca
- ReceiveOpenString
  - Arguments: ep: LongInt; err: LongInt; command_line: String
  - Documentation: Recebe pedido de Inicialização do Kernel do Mathematica
- SendOpenString
  - Arguments: result: LongInt; ep: LongInt; errp: LongInt
  - Documentation: Envia resposta da inicialização do Kernel
- ReceiveMLConnect
  - Arguments: mlp: LongInt
  - Documentation: Recebe pedido de conexão com o Kernel do Mathematica
- SendMLConnect
  - Arguments: mlp: LongInt; res: Integer
  - Documentation: Envia resposta de conexão com o Kernel
- ReceiveMLPutFunction
  - Arguments: mlp: LongInt; argc: LongInt; s: String
- ReceiveMLPutString
  - Arguments: mlp: LongInt; s: String
- MLDoCompute
  - Return Class: Integer

Classe: **TGenComunic**
Descrição: Gerente de Comunicação.
Atributos:

parent : TgenComport; Trad_AV : TTradutor_AV; Trad_FE_port : Integer; Log : TregOcor; AV_connect : Boolean; FE_connect : Boolean; proTrad_FE : TbioProtocol; ceTrad_FE : TCommunicationElementClientBIO; ep : Integer link : Integer; err : Integer; sPath : String; initstring : String; InfoMsg : String

Métodos:

- SendMsgToFE
  - Arguments: Msg: String
  - Documentation: Envia mensagens de texto para a Ferramenta Externa.
- ReceiveAccept
  - Arguments: Login: String; Result: char
  - Documentation: Recebe Mensagem de Accept do Tradutor da FE
- SendMLRequestInitialize
  - Arguments: Integer p
  - Documentation: Envia mensagem de requisição de inicialização da biblioteca do Mathematica
- ReceiveMLInitialize
  - Arguments: Result: Integer
  - Documentation: Recebe mensagem de inicialização da biblioteca do Mathematica
- SendMLRequestOpenString
  - Arguments: ep: Integer; errp: Integer; command_line: String
  - Documentation: Envia mensagem de requisição de inicialização do Kernel do Mathematica
- ReceiveMLOpenString
  - Arguments: result: Integer; ep: Integer; errp: Integer; command_line: String
  - Documentation: Recebe mensagem de inicialização do Kernel do Mathematica
- SendMLRequestConnect
  - Arguments: mlp: Integer
  - Documentation: Envia mensagem de requisição de conexão com o Kernel do Mathematica
- ReceiveMLConnect
  - Arguments: mlp: Integer; res: Integer
  - Documentation: Recebe mensagem de conexão com o Kernel do Mathematica
- SendMLPutFunction
  - Arguments: mlp: Integer; argc: Integer; s: String
  - Documentation: Envia mensagem contendo uma função para o Kernel do Mathematica
- SendMLPutString
  - Arguments: mlp: Integer; s: String
  - Documentation: Envia mensagem contendo uma string para o Kernel do Mathematica
- ReceiveMLReturnTextPkt
  - Arguments: mlp: Integer; s: String
  - Documentation: Recebe mensagem contendo um pacote de retorno em texto
- ReceiveFEDisconnect

Documentation: Recebe mensagem de desconexão do Tradutor da FE

SendTxtToAV
Arguments: iUser: Integer; Msg: String
Documentation: Envia mensagens de texto para o usuário identificado por iUser do Ambiente Virtual.

SendURLToAV
Arguments: iUser: Integer; Urls: TStringList
Documentation: Envia endereços Internet com suas respectivas descrições para o usuário identificado por iUser do Ambiente Virtual.

SendTurn
Arguments: angle: Integer
Documentation: Vira a BIA na angulação indicada por angle

Create
Return Class: TGenComunic
Arguments: pParent: TGenComport

ReceiveMsg
Arguments: iSender: Integer; SenderNum: Integer; Msg: String
Documentation: Procedimento chamado ao chegar mensagens para a Bia.

IsConnect
Documentation: Procedimento chamado para detectar se a BIa está conectada com as duas extremidades.

ReceiveMLTextPkt
Arguments: mlp: Integer; s: String

ReceiveMLDisplayEndPacket
Arguments: mlp: Integer; s: String

ReceiveMLMessagePkt
Arguments: mlp: Integer; s: String

SendGraphicToAV
Arguments: User: Integer; filename: String
Documentation: Envia mensagem avisando que o arquivo de imagem está disponível no servidor de ftp.

SendGen
Arguments: User: Integer; Msg: String

ReceiveGen
Arguments: User: Integer; Msg: String

Classe: **TRegOcor**
Descrição: Registrador de Ocorrências
Atributos:
logfile : TextFile; pComport : TgenComport; pComunic : TGenComunic
Métodos:

Create
Return Class: TRegOcor
Arguments: pCp: TgenComport; pCm: TGenComunic

WriteToLog
Arguments: Text: String
Documentation: Escreve linha no arquivo de log.

OpenLog
Return Class: boolean
Arguments: Filename: String

Documentation: Abre o arquivo de log.

CloseLog

Documentation: Fecha o arquivo de log.

Classe: **TGenSimul**
Descrição: Gerente de Simulóide.
Atributos:

pos_x : Integer; pos_y : Integer; pos_z : Integer; IniFile : TiniFile; State : Integer; Actions : Tlist; parent : TGenComport

Métodos:

Create

Return Class: TGenSimul
Arguments: pParent: TGenComport

DoAction

Return Class: TList
Arguments: action: String
Documentation: Dada a ação action, retorna uma lista de comandos correspondentes.

SetState

Arguments: iState: Integer
Documentation: Configura o estado da BIA.

GetPos

Return Class: TPoint
Documentation: Retorna as coordenadas x e y da Bia no Ambiente Virtual.

SetPos

Arguments: posX: Integer; posY: Integer; posZ: Integer
Documentation: Configura as posições x, y e z da BIA no Ambiente Virtual.

LoadFromIniFile

Return Class: TList
Arguments: IniFile: TIniFile
Documentation: Carrega a Lista de Ações do arquivo.

WhatTimeIs

Return Class: String
Documentation: Função que retorna a parte do dia do Ambiente Virtual: M: Manhã; T:Tarde e N: Noite.

Classe: **TRequisition**
Atributos:

iUser : Integer (Número do Usuário que fez a requisição); sReq : String (Requisição); sMathReq : String (requisição no formato do Mathematica); iFunction : Integer (Índice da função em BeremizFunctions); SendAll : Boolean

Métodos:

Create

Return Class: TRequisition
Arguments: iUser: Integer; sReq: String; iType: Integer; iFunction: Integer; sMathReq: String; SendAll: Boolean

Classe: **TGenComport**
Descrição: Gerente de Comportamento
Atributos:

Simuloide : TgenSimul; Comunicacao : TgenComunic; Conhecimento : TgenConhec; Log : TregOcor; UserList : TstringList; RoomUserList : Array of boolean

Métodos:

SendText
Arguments: iDestiny: Integer; iUser: Integer; Msg: String
Documentation: Envia mensagens para o Controlador de Comunicação, que as envia para o usuário iUser da extremidade identificada por iDestiny.

SendExprToFE
Arguments: Msg: String
Documentation: Envia Mensagem para a FE

SendURL
Arguments: iUser: Integer; Urls: TStringList
Documentation: Envia endereços internet para o usuário identificado por iUser do ambiente virtual.

ProcessMessage
Arguments: iSender: Integer; SenderNum: Integer; Msg: String
Documentation: Trata todas as mensagens que chegam para o Beremiz.

ProcessAction
Arguments: List: Tlist; iUser: Integer
Documentation: Processa as ações recebidas do controlador de simuloide enviando os respectivos tipos de mensagem para o ambiente virtual.

Create
Return Class: TGenComport

VerPos
Arguments: SenderNum: Integer; PosX: Integer; PosY: Integer
Documentation: Verifica se o avatar está próximo do Beremiz

DelUserRoom
Arguments: iUser: Integer
Documentation: Deleta usuário identificado por iUser da lista de usuários que estão na sala de matemática.

SendTurn
Arguments: angle: Integer
Documentation: Envia mensagem de rotação para o ambiente virtual.

Destroy

ReceiveMLReturnTextPkt
Arguments: mlp: Integer; s: String
Documentation: Recebe mensagem contendo um pacote de retorno em texto da FE

ReceiveMLTextPkt
Arguments: mlp: Integer; s: String

ReceiveMLDisplayEndPacket
Arguments: mlp: Integer; s: String

ReceiveMLMessagePkt
Arguments: mlp: Integer; s: String

SendGraphicToAV
Arguments: User: Integer; filename: String

Classe: **TParser**
Descrição: Classe parser

Atributos:
parent : TgenConhec; sName : String; sSource : String; Pattern : TStringList;
keyword : Boolean; iFuncMath : Integer; iFuncBeremiz : Integer
Métodos:
Create
Return Class: TParser
Arguments: parent: TgenConhec; sSource: String
GetFirst
Return Class: String
Arguments: sSource: String
GetNext
Return Class: String
Arguments: sSource: String; sNext: String
Parser
Return Class: Integer
Arguments: sSource: String; Pattern: TStringList; Args: TList
IsKeyword
Return Class: Boolean
Arguments: sName: String; Keywords: TStringList

Classe: **TGenConhec**
Descrição: Gerente de Conhecimento.
Atributos:
ReqList : Tlist; parent : TgenComport; ReqTypes : Tlist; Parser : Tparser;
FunctionSet : TfunctionSet; KeyWords : TstringList; ReqTxtList : Tlist;
ReqGraphList : TList
Métodos:
Create
Return Class: TGenConhec
Arguments: pparent: TGenComport
Documentation: construtor.
VerifyRequisition
Return Class: Integer
Arguments: Msg: String; List: TList
Documentation: Verifica o tipo de requisição.
VerifyResult
Return Class: Integer
Arguments: Msg: String; List: TList
Documentation: Verifica e processa o tipo de retorno da FE
ProcessRequisition
Return Class: String
Arguments: SenderNum: Integer; Data: TStringList; iType: Integer
Documentation: Processa a requisição
SendExprToFE
Return Class: String
Arguments: Msg: String
Documentation: Formata Mensagem para o Controlador de Comportamento
para ser repassada para a FE
SendGraphic
Arguments: FileName: String

ProcessMLMessagePkt
Arguments: mlp: Integer; s: String
GetMathFunctions
Return Class: TStringList
GetBeremizFunctions
Return Class: TStringList
CheckMathName
Return Class: Integer
Arguments: sName: String
CheckFuncName
Arguments: sName: String
InsertMathFunction
Arguments: sName: String; iArg: Integer; ArgsType: TStringList
EditMathFunction
Arguments: index: Integer; iArg: Integer; ArgsType: TStringList
DeleteMathFunction
Arguments: index: Integer
InsertBeremizFunction
Arguments: sName: String; iType: Integer; sMathName: String; sPattern: String; SendAll: Boolean
EditBeremizFunction
Arguments: index: Integer; iType: Integer; sMathName: String; sPattern: String; SendAll: Boolean
DeleteBeremizFunction
Arguments: index: Integer
GetMathDetails
Return Class: TStringList
Arguments: sName: String
GetMathData
Return Class: TStringList
Arguments: sName: String
GetBeremizData
Return Class: TStringList
Arguments: sName: String
GetPattern
Return Class: TStringList
Arguments: index: Integer
DelExtraBlanks
Return Class: String
Arguments: s: String
GetReqTypes
Return Class: TStringList
GetIType
Return Class: Integer
Arguments: sType: String
ChangeParenthesis
Return Class: String
Arguments: sSource: String
Documentation: função que muda os '( )' numa chamada de função de uma expressão para '[ ]'.

Change_Parenthesis
Return Class: String
Arguments: sSource: String
Documentation: função que muda os '( )' numa chamada de função de uma expressão para '[ ]', acrescentando o '_' depois da (s) variável(is) de dentro dos '[ ]'
IsIdent
Return Class: Boolean
Arguments: sSource: String
Documentation: função que detecta se a string sSource é alfanumérica
GetClosePar
Return Class: Integer
Arguments: s: String; i: Integer
Documentation: função que retorna a posição do ')' correspondente ao '(' da posição i.
GetAllArgTypes
Return Class: TStringList
Documentation: função que retorna todos os nomes dos tipos dos argumentos das funções
CheckArgName
Return Class: Integer
Arguments: sName: String
CheckArgOutName
Return Class: Integer
Arguments: Name: String
InsertArgType
Arguments: sName: String; iNum: Integer; Types: TStringList; sOutName: String; chLeft: String; chRight: String; chSep: String
EditArgType
Arguments: index: Integer; iNum: Integer; Types: TStringList; sOutName: String; chLeft: String; chRight: String; chSep: String
DeleteArgType
Arguments: index: Integer
GetBasicArgTypes
Return Class: TStringList

Classe: **TFunctionMath**
Descrição: Classe de função do Mathematica
Atributos:
sName : String (Nome da função); iArg : Integer (Número de argumentos); ArgsType : TstringList(Tipos dos argumentos (nomes do tipo TArgType))

Métodos:
Create
Return Class: TFunctionMath
Arguments: sName: String; iArg: Integer; ArgsType: TStringList
Documentation: Construtor
SaveToFile
Arguments: IniFile: TIniFile
Documentation: Salva dados das funções no arquivo de nome IniFile

Destroy
FormatFunction
Return Class: String
Arguments: Listpar: TList
Documentation: Formata a função para o Mathematica utilizando os parâmetros passados.

Classe: **TFunctionBeremiz**
Descrição: Função mapeada para a linguagem do Beremiz
Atributos:
sName : String (nome da função); iType : Integer (Tipo da requisição);SendAll : Boolean (Sinalizador para enviar para todos os usuários); sMathName : String (Nome da função correspondente no Mathematica); sPattern : String (Padrão de sentença de reconhecimento)
Métodos:
Create
Return Class: TFunctionBeremiz
Arguments: sName: String; sMathName: String; sPattern: String; argname: Integer; SendAll: Boolean
Documentation: Construtor
SaveToFile
Arguments: IniFile: TIniFile
Documentation: Salva os dados das funções no arquivo de nome IniFile
Destroy

Classe: **TFunctionSet**
Descrição: Conjunto de funções
Atributos:
MathFile : TiniFile; FuncFile : TiniFile; ArgsFile : TiniFile; MathFunctions : Tlist; BeremizFunctions : Tlist; ArgTypes : TList
Métodos:
Create
Return Class: TFunctionSet
Destroy
LoadMathFunctions
Return Class: TList
Arguments: IniFile: TIniFile
Documentation: Carrega os dados sobre as funções do Mathematica cadastradas no arquivo.
LoadBeremizFunctions
Return Class: TList
Arguments: IniFile: TIniFile
Documentation: Carrega os dados sobre as funções do Beremiz cadastradas no arquivo.
LoadArgTypes
Return Class: TList
Arguments: IniFile: TIniFile
Documentation: Carrega os dados sobre os tipos de argumentos que o Mathematica recebe.
GetMathFunctions

Return Class: TStringList
Documentation: retorna todos os nomes das funções do Mathematica.

GetBeremizFunctions
Return Class: TStringList
Documentation: retorna todos os nomes das funções do Beremiz.

GetAllArgTypes
Return Class: TStringList
Documentation: retorna todos os nomes dos tipos dos argumentos cadastrados.

GetBasicArgTypes
Return Class: TStringList
Documentation: retorna todos os nomes dos tipos base dos argumentos cadastrados.

CheckMathName
Return Class: Integer
Arguments: sName: String
Documentation: Verifica se o nome da função do Mathematica já está cadastrada.

CheckFuncName
Return Class: Integer
Arguments: sName: String
Documentation: Verifica se o nome da função do Beremiz já está cadastrada.

CheckArgName
Return Class: Integer
Arguments: sName: String
Documentation: Verifica se o nome interno do tipo do argumento já está cadastrado.

CheckArgOutName
Return Class: Integer
Arguments: sName: String
Documentation: Verifica se o nome de interface do tipo do argumento já está cadastrado.

InsertMathFunction
Arguments: sName: String; iArg: Integer; ArgsType: TStringList
Documentation: Insere no arquivo uma nova função do Mathematica

EditMathFunction
Arguments: index: Integer; iArg: Integer; ArgsType: TStringList
Documentation: Altera os dados no arquivo da função do Mathematica

DeleteMathFunction
Arguments: index: Integer

InsertBeremizFunction
Arguments: sName: String; iType: Integer; sMathName: String; sPattern: String; SendAll: Boolean

EditBeremizFunction
Arguments: index: Integer; iType: Integer; sMathName: String; sPattern: String; SendAll: Boolean

DeleteBeremizFunction
Arguments: index: Integer

InsertArgType

Arguments: sName: String; iNum: Integer; Types: TStringList; sOutName: String; chLeft: String; chRight: String; chSep: String

EditArgType
Arguments: index: Integer; iNum: Integer; Types: TStringList; sOutName: String; chLeft: String; chRight: String; chSep: String

DeleteArgType
Arguments: index: Integer

GetMathDetails
Return Class: TStringList
Arguments: sName: String

GetMathData
Return Class: TStringList
Arguments: sName: String

GetBeremizData
Return Class: TStringList
Arguments: sName: String

GetPattern
Return Class: TStringList
Arguments: index: Integer

CreateArg
Return Class: TArg
Arguments: index: Integer; data: TStringList

GetArgOutName
Return Class: String
Arguments: sName: String
Documentation: Dado o nome interno, retorna o nome de interface.

GetArgInName
Return Class: String
Arguments: sOutName: String
Documentation: Dado o nome de interface, retorna o nome interno

Classe: **TArgType**
Descrição: Classe do tipo de argumento
Atributos:
sName : String (nome interno do tipo); iNum : Integer (número de elementos); Types : TstringList (tipos dos elementos); sOutName : String (nome que aparece nos formulários); chLeft : String (caractere delimitador esquerdo); chRight : String (caractere delimitador direito); chSep : String (caractere separador)
Métodos:
Create
Return Class: TArgType
Arguments: sName: String; iNum: Integer; Types: TStringList; sOutName: String; chLeft: String; chRight: String; chSep: String
Documentation: construtor

Destroy

SaveToFile
Arguments: IniFile: TIniFile
Documentation: Salva os dados dos tipos no arquivo IniFile

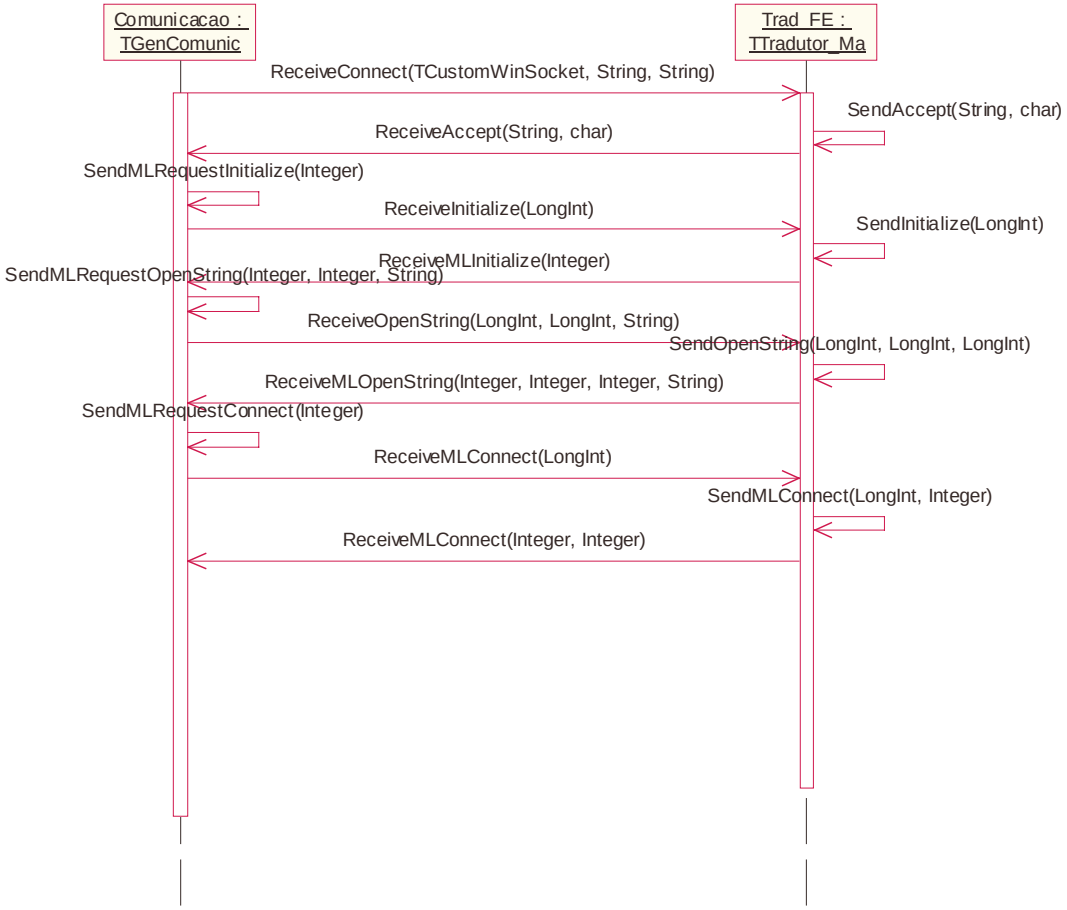
GetchLeft
Return Class: String

GetchRight
Return Class: String
GetchSep
Return Class: String

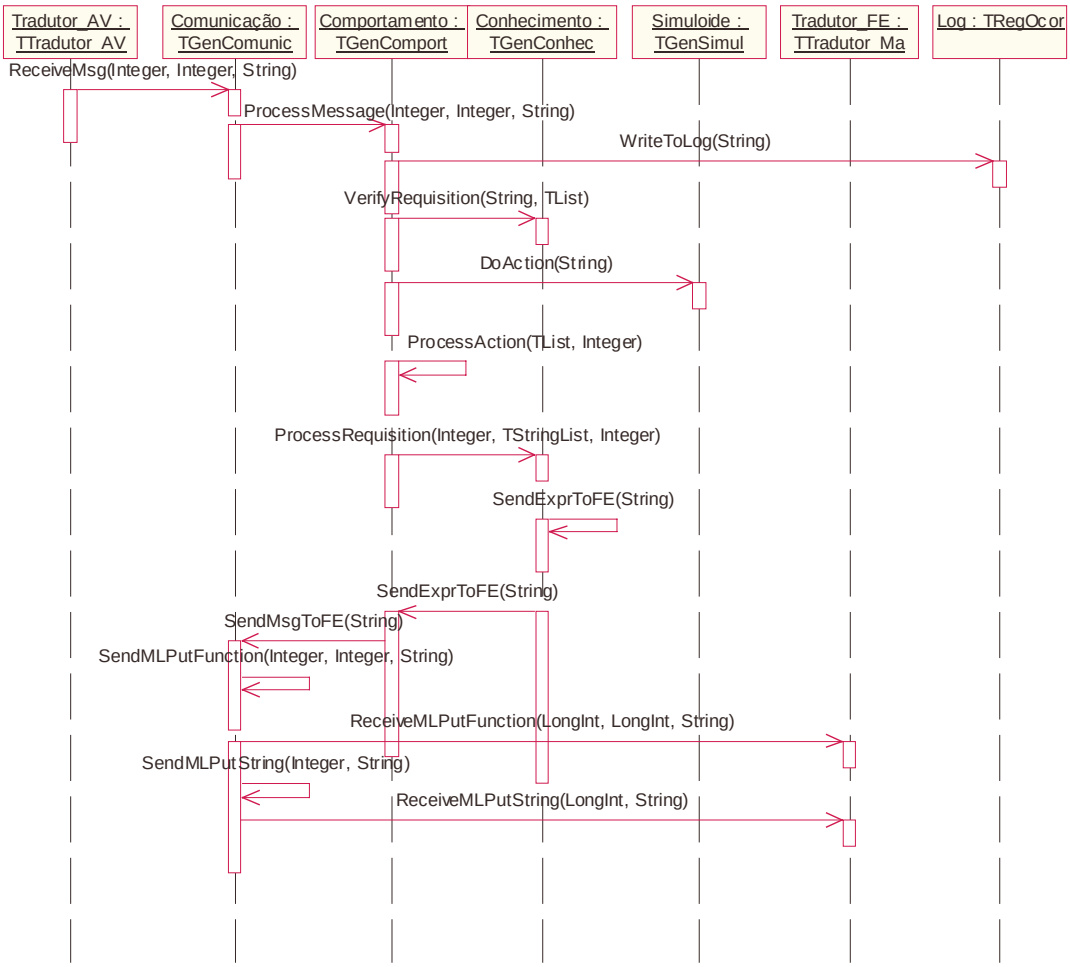
Classe: **TArg**
Descrição: classe de argumento
Atributos:
argType : TargType; data : TStringList
Métodos:
Create
Return Class: TArg
Arguments: argType: TargType; data: TStringList
FormatFunction
Return Class: String

Anexo 7

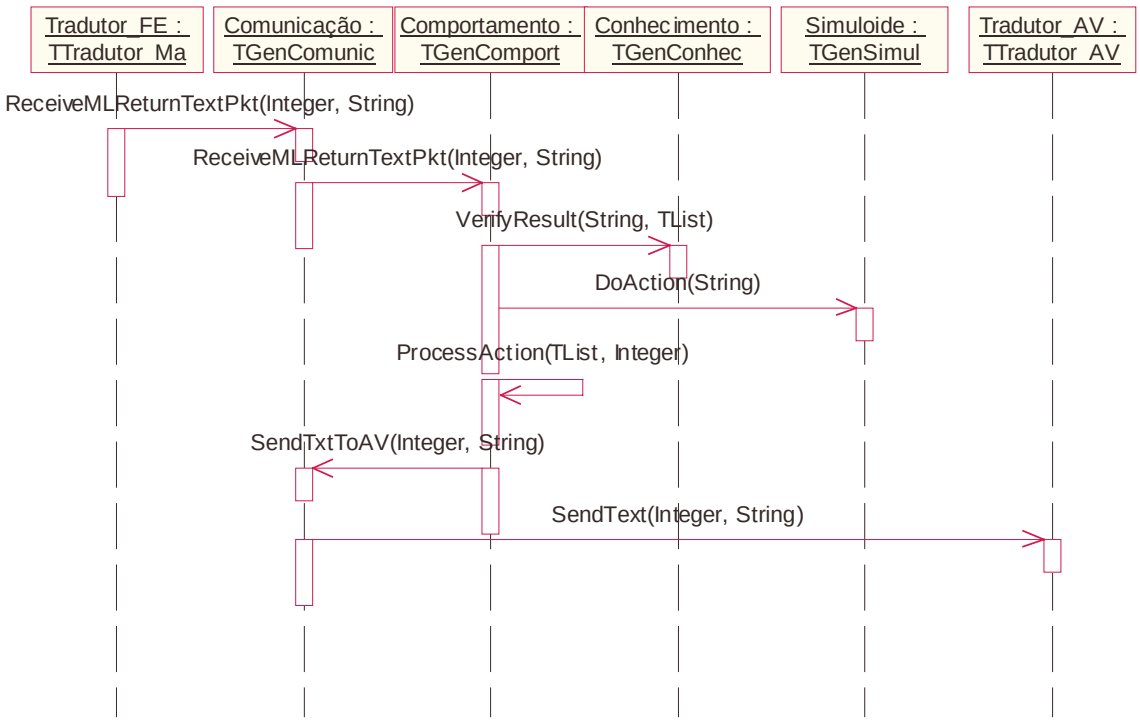
Diagrama de Seqüência da Aplicação Beremiz



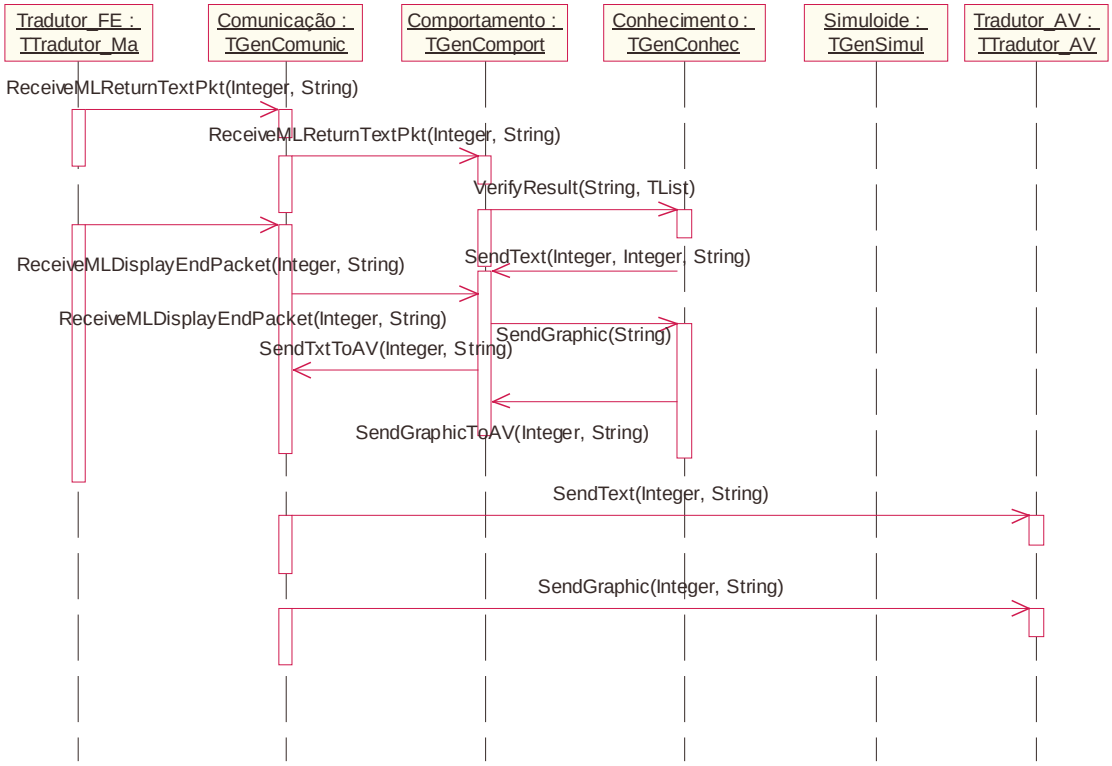
Cenário: Conexão com o tradutor do Mathematica e inicialização da biblioteca mathlink.



Cenário: Trajeto de requisição do ambiente virtual ao Mathematica.



Cenário: Trajeto de resposta de requisição do tipo calcule ou defina do Mathematica ao ambiente virtual.



Cenário: Trajeto de resposta de requisição do tipo desenhe do Mathematica ao ambiente virtual

Anexo 8

Arquivo de Script de Ações do Simulóide Controlado pela Aplicação Beremiz

```
;; Arquivo de inicialização do controle de simulóide do Beremiz
;; Os comandos devem ser colocados na ordem em que eles serão executados
;; Posteriormente colocar o tipo de mensagem a que eles correspondem
;; O primeiro caractere depois do = definirá o tipo:
;; T: mensagem de texto
;; M: movimentação do simulóide (ação)
;; S: som
;; E: expressão facial
;; D: deslocamento

;Saudação Inicial Manhã
[SIM]
0=D,Virar para usuário
1=E,sorrindo
2=T,Bom Dia!
3=S,bomdia.mp3
4=T,Posso Ajudá-lo?
5=E,normal

;Saudação Inicial Tarde
[SIT]
0=D,Virar para usuário
1=E,sorrindo
2=T,Boa Tarde!
3=S,boatarde.mp3
4=T,Posso Ajudá-lo?
5=E,normal

;Saudação Inicial Noite
[SIN]
0=D,Virar para usuário
1=E,sorrindo
2=T,Boa Noite!
3=S,boanoite.mp3
4=T,Posso Ajudá-lo?
5=E,normal

;Requisição de usuário
[REQ]
0=E,sorriso enigmático
1=T,Aguarde um instante, por favor...

;Requisição não compreendida
[REQINCOMP]
0=E,desconsertado
1=S,perdao.mp3
2=T,Desculpe, não entendi a sua pergunta!

;Posição Normal
[NORMALPOS]
0=E,normal

;Resposta de Cálculo
[RESPCALC]
0=E,sorrindo
1=S,acheidesc.mp3

;Resposta de Definição
[RESPDEF]
0=E,sorrindo
1=S,defok.mp3
```

Anexo 9

Arquivos de informações de mapeamentos de funções da Aplicação Beremiz

Arquivo com os dados das funções na linguagem do usuário.

```
;; Arquivo de inicialização dos nomes das funções no beremiz
;; [] -> nome da função
;; 0 -> tipo da requisicao
;; 1 -> SendAll?
;; 2 -> Função no Mathematica
;; 3 -> String de entrada

[desenhetodos]
0=6
1=True
2=Plot
3=desenhetodos _listafunc com _var entre _cte e _cte

[desenhe3d]
0=6
1=True
2=Plot3D
3=desenhe3d _funcao com {_var,_cte,_cte} e {_var,_cte,_cte}

[minlocal]
0=5
1=False
2=FindMinimum
3=minlocal _funcao partindo de _var = _cte
```

Arquivo com os dados das funções do Mathematica.

```
;; Arquivo de inicialização dos nomes das funções
;; [] -> nome da função no mathematica
;; 0 -> número de argumentos
;; 1 -> tipos dos argumentos separados por ","


[Plot]
0=2
1=_listafunc,_intervalo3
[Plot3D]
0=3
1=_funcao,_intervalo3,_intervalo3
[FindMinimum]
0=2
1=_funcao,_intervalo2
```

Arquivo com os dados dos tipos dos argumentos.

```
;; Arquivo de inicialização dos tipos dos argumentos das funções
;; [] -> nome interno do tipo
;; 0=número de elementos
;; 1=tipo dos elementos
;; 2=nome que deve aparecer
;; 3=caractere delimitador esquerdo
;; 4=caractere delimitador direito
;; 5=caractere separador

[_var]
0=1
1=
2=var
3=
4=
5=

[_cte]
0=1
1=
2=cte
3=
4=
5=


[_intervalo2]
0=2
1=_var,_cte
2=Intervalo (2)
3={
4=}
5=,

[_intervalo3]
0=3
1=_var,_cte,_cte
2=Intervalo (3)
3={
4=}
5=,

[_listafunc]
0=-1
1=_var
2=Lista de Funções
3={
4=}
5=,

[_funcao]
0=1
1=_var
2=função
3=
4=
5=
```