UNIVERSIDADE FEDERAL DO RIO GRANDE DO NORTE
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

GATE - Uma abordagem baseada em middleware para aplicações interperceptivas envolvendo múltiplos dispositivos

Rummenigge Rudson Dantas

Orientador: Prof. Dr. Luiz Marcos G. Gonçalves

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia Elétrica e de Computação da UFRN (área de concentração: Engenharia de Computação) como parte dos requisitos para obtenção do título de Doutor em Ciências.

Número de ordem PPgEE: D53
Natal, RN, Janeiro de 2009

Divisão de Serviços Técnicos

Catalogação da publicação na fonte. UFRN / Biblioteca Central Zila Mamede

Rummenigge Rudson Dantas.

GATE - Uma abordagem baseada em middleware para aplicações interperceptivas envolvendo múltiplos dispositivos / Rummenigge Rudson Dantas - Natal, RN, 2010

23 p.

Prof. Dr. Luiz Marcos G. Gonçalves

Tese (doutorado) - Universidade Federal do Rio Grande do Norte. Centro de Tecnologia. Programa de Pós-Graduação em Engenharia Elétrica.

1. Redação técnica - Tese. 2. LaTeX- Tese. I. Dantas, Rummenigge Rudson. II. Gonçalves, Luiz Marcos Garcia. III. Título.

RN/UF/BCZM CDU 004.932(043.2)

GATE - Uma abordagem baseada em middleware para aplicações interperceptivas envolvendo múltiplos dispositivos

Rummenigge Rudson Dantas

Tese de Doutorado aprovada em 18 de janeiro de 2010 pela banca examinadora composta pelos seguintes membros:

Prof. Dr. Luiz Marcos Garcia Gonçalves (Orientador) ..... UFRN

Prof. Dr. Luiz Eduardo Cunha Leite ..... UFRN

Prof. Dr. Aquiles Medeiros Filgueira Burlamaqui ..... UFRN

Prof. Dr. Alejandro César Frery Orgambide ..... UFAL

Prof[a]. Dr[a]. Tatiana Aires Tavares ..... UFPB

Prof[a] Dr[a] Eliana Silva de Almeida ..... UFAL

Aos meus pais, Assis e Francisca e à toda família Natalnet. Obrigado por tudo.

Agradecimentos

Ao meu orientador, professor Luiz Marcos, sou grato pela orientação.

Ao professor Aquiles, pelas sugestões e várias colaboraçãoes ao longo desta tese.

À todas as pessoas que fazem a Natalnet, por sua participação e empenho nos vários projetos de pesquisa que ajudaram a tornar essa tese o que ela é hoje.

Aos demais colegas de pós-graduação, pelas críticas, sugestões e pelas partidas de DOTA.

À todos os amigos que colaborarm, apoiaram ou torceram por mim nesta longa caminhada.

À minha família pelo apoio durante esta jornada e por fazer de mim a pessoa que sou hoje.

Ao CNPQ, pelo apoio financeiro.

Resumo

Neste trabalho, apresentamos o GATE, uma abordagem baseada em *middleware* para aplicações interperceptivas. Através dos serviços oferecidos pelo GATE, estendemos o conceito da Interpercepção para a integração com vários dispositivos, incluido set-top box, dispositivos móveis (celulares), entre outros. Através desta extensão garantimos a execução de ambientes virtuais nesses dispositivos. Assim, usuários que acessarem a versão do ambiente pelo computador poderão interagir com aqueles que acessarem o mesmo ambiente por outros dispositivos. Essa extensão é apenas uma parte dos serviços garantidos pelo GATE, surge como uma nova proposta para criação de ambientes virtuais multisuários.

Palavras-chave: Sistemas Distribuídos, Middleware, Interpecepção, Protocolos de Comunicação.

Abstract

In this work, we present the GATE, an approach based on middleware for interperceptive applications. Through the services offered by the GATE, we extension we extend the concept of Interperception for integration with several devices, including set-top box, mobile devices (cell phones), among others. Through this extension ensures the implementation of virtual environments in these devices. Thus, users who access the version of the computer environment may interact with those who access the same environment by other devices. This extension is just a part of the services provided by the GATE, that remerges as a new proposal for multi-user virtual environments creation.

Keywords: Distributed Systems, Middleware, Interpeception, Communication Protocols.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Novos dispositivos com capacidade computacional surgem a cada dia. A maioria desses aparelhos possui a capacidade de conectar-se a alguma rede. Essa conexão permite que esses dispositivos executem aplicações que integram vários usuários. Através dessa integração os usuários compartilham informações. As aplicações que permitem esse compartilhamento são chamadas de aplicação multiusuário. Essas aplicações são muito importantes para as comunicações atuais. A interação promovida pelas mesmas mudou o curso da comunicação humana. Ferramentas de bate-papo e ambientes virtuais multiusuário são alguns exemplos dessas aplicações.

Um dos primeiros exemplos de ambientes virtuais multiusuário da história foi proposto no final da década de 1970. Roy Trubshaw e Richard Bartle, dois pesquisadores da Universidade de Essex, desenvolveram um *software* capaz de simular um mundo virtual por simples descrição textual [Bartle 1990]. O jogo desenvolvido pelos pesquisadores de Essex combinava elementos de RPG [Fine 2002] e interação social através de salas de bate-papo (*chat*). MUD (*Multi-user Dungeon*) [Wolf 2008] como ficou batizado o jogo, era executado através de uma BBS (*Bulletin Board System*) o que permitia a interação entre usuários remotamente localizados. Seus jogadores liam a descrição de salas, objetos, eventos e outros personagens. Cabia ao controle do software determinar as ações dos NPC's (*Non-Player Characters*). Os jogadores podiam interagir entre si e com o ambiente através de comandos que se assemelham à linguagem natural, normalmente em inglês. A Figura 1.1 mostra a interface comum de um MUD.

Nas décadas seguintes à criação do MUD, surgiram ambientes multiusuário com interfaces de visualização 2D e depois 3D. Assim como o MUD, o foco estava na utilização dos mesmos para criação de jogos. Contudo, um dos exemplos com interface de visualização 3D foi desenvolvido para ser um navegador da rede: *Active Worlds* [Ensor 2003]. Na época do nascimento do *Active Worlds*, as redes de computadores convergiam para a *Internet*. A rede mundial de computadores se tornou o canal para o crescimento e maturidade dos aplicativos multiusuário, inclusive os ambientes virtuais.

Apesar do esforço para se criar interfaces com gráficos cada vez mais avançados, ainda é possível encontrar ambientes virtuais, na *Internet*, com interfaces textuais (http://mud.valinor.com.br/), interfaces 2D (http://iro.ragnarokonline.com/) e interfaces 3D (http://www.runescape.com/). Essa situação pode ser explicada pelo fato de que as interfaces de visualização 3D necessitam de mais recursos (de *hardware* e *software*) que as interfaces

Figura 1.1: Interface de um MUD.

com elementos gráficos 2D e textuais. A quantidade de processamento computacional necessário para exibição de gráficos 3D é maior que o processamento para exibição dos demais gráficos. Nem todos os usuários possuem os recursos mínimos para realizar tal processamento. Por isso, esses usuários procuram ambientes que exijam menos recursos. Isto também significa buscar ambientes com gráficos menos custosos para realizar o processamento.

A disparidade de perfis de usuários que utilizam ambientes virtuais aponta para uma segregação entre os que possuem recursos (de *hardware* e *software*) mais avançados e os que possuem recursos menos avançados. Por exemplo, os que possuem os recursos mais avançados podem acessar os mais novos ambientes 3D com gráficos detalhados. Do outro lado, aqueles com menos requisitos devem se contentar com outro ambiente interface de visualização com gráficos menos complexos (como 2D, por exemplo). Contudo, todos esses usuários possuem um meio comum para o acesso aos ambientes: a *Internet*.

Mesmo que o meio usado para acesso seja o mesmo, a separação desses usuários é justificável pelo fato de cada um desses ambientes virtuais representar um conjunto diferente de informações. Se as informações que caracterizam um determinado ambiente *A* pudessem ser representadas com diferentes visualizações gráficas (ou seja, com gráficos 3D, 2D ou mesmo texto) seria possível criar diferentes aplicativos para o acesso do ambiente *A*. Mesmo que cada um desses aplicativos possua uma interface de visualização distinta, eles ainda representam o mesmo ambiente. Se tivermos o mesmo ambiente sendo representado é possível tentar estabelecer uma ligação entre as diferentes aplicações usadas para acessá-lo. Ao estabelecer essa ligação, os usuários com diferentes perfis (e recursos) poderiam fazer parte de um mesmo ambiente virtual compartilhado *on-line*.

Baseado nas suposições do parágrafo anterior foi proposto o conceito de *Interpercepção* [Azevedo et al. 2006]. A proposta da *Interpercepção* é permitir a criação de ambientes

virtuais com diferentes opções de interface de visualização. Cada opção é representada por um aplicativo diferente. A *interpercepção* fornece também meios para permitir a interligação desses aplicativos através da rede. A interligação dos aplicativos permite que todos os usuários que acessam o ambiente virtual num dado instante sejam capazes de interagir com os demais, independente da versão usada para visualização. A *Interpercepção* é apresentada de forma mais detalhada a seguir.

1.1 *Interpercepção*

O funcionamento primordial da *Interpercepção* é baseado na idéia apresentada na Figura 1.2. Nessa figura temos que um ambiente virtual *A* pode ser representado por um arquivo de meta-dados *M*. Esse conjunto de meta-dados é armazenado em um banco de dados. Esse banco faz parte de um sistema distribuído. Esse sistema também conta com *n* aplicativos que podem acessar remotamente o banco.

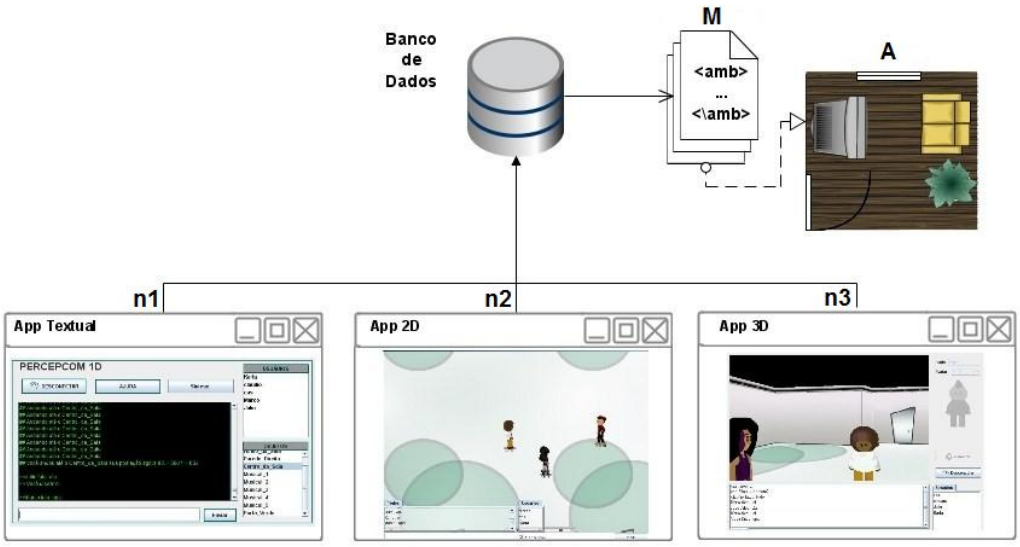

Figura 1.2: Cenário de funcionamento da Interpercepção.

Os aplicativos são capazes de ler o arquivo de meta-dados que representa o ambiente e transcrevê-los para se adequar à sua interface de visualização. Assim, uma dada aplicação *App 2D* (que possui interface de visualização 2D) depois de ler o arquivo de meta-dados irá converter essa informação em gráficos 2D.

Na Figura 1.2 temos outros dois aplicativos (*App textual* e *App 3D*) que transcrevem a informação do arquivo de meta-dados para a interface suportada pelo aplicativo. O processo de transcrição deve ser estabelecido para cada aplicativo. Esse processo é feito por uma rotina que converte meta-dados em estruturas adequadas a cada tipo de aplicativo.

O retrospecto aqui apresentado sobre a *Interpercepção* é voltado para sua utilização em sistemas baseados em computadores pessoais (abreviado daqui em diante para PC, personal computer). Contudo, outros dispositivos além do PC (com um sistema embarcado [Ganssle 2007]) permitem a conexão com a *Internet*. Os celulares, por exemplo, já

permitem conexão com a *Internet* há um bom tempo. Através da extensão do conceito de *Interpercepção* para celulares, por exemplo, será possível uma aplicação para este tipo de dispositivo ser executada através de diferentes interfaces de visualização. Além disto se este mesmo aplicativo possuir uma versão para *desktop*, seria possível integrar a versão para celular com versão para PC e criar um ambiente colaborativo e compartilhado entre os usuários de todas as versões. Esse cenário é ilustrado na Figura 1.3.

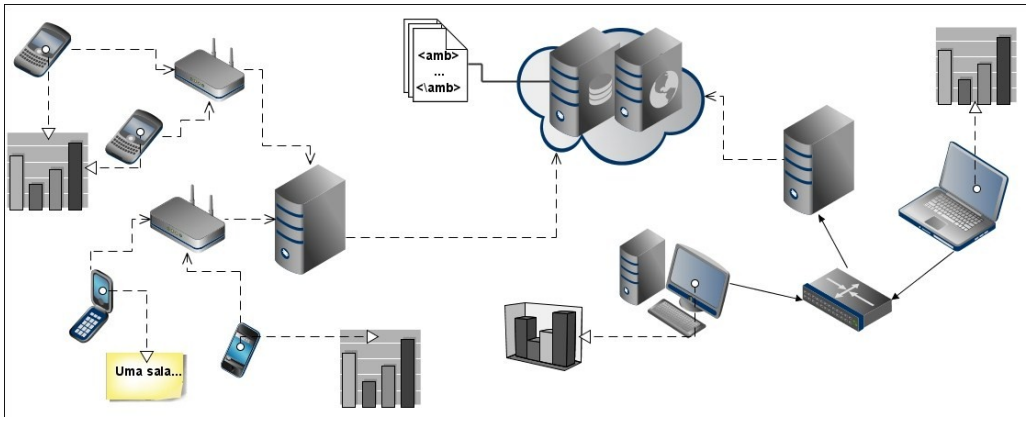

Figura 1.3: Cenário da Interpercepção para vários dispositivos

No exemplo da Figura 1.3 temos um conjunto de arquivos de meta-dados armazenados em uma nuvem de servidores. Esses arquivos representam um ambiente virtual criado com base nas leis da *Interpecepção*. Uma rede de celulares acessa um roteador *wireless*. O roteador está conectado com a nuvem de servidores. Os celulares agora interligados em rede executam o ambiente virtual multiusuário armazenado na nuvem. A rede de celulares pode também estabelecer comunicação com uma rede de PCs que está conectada a mesma nuvem de servidores. Isso daria início a uma grande rede de dispositivos *on-line*. Os dispositivos nessa rede poderiam então compartilhar informações de um mesmo ambiente virtual em *on-line*.

A integração entre vários dispositivos diferentes caracteriza o surgimento de um sistema distribuído *pervasivo/ubíquo* [Weiser 1991*a*]. Um sistema pervasivo caracteriza-se por permitir o acesso à computação em todo lugar, o tempo todo. Para a criação de tal sistema, os dispositivos envolvidos devem ser capazes de estabelecer comunicação independente de protocolos de comunicação, plataformas e dependência do sistema operacional. O estabelecimento dessa comunicação é possível através da utilização de *middlewares* [Vinoski 2004, Bernstein 1996].

Middlewares são camadas de *software* que fazem a mediação entre outros *softwares* (ou mesmo *hardwares*). Os sistemas de *middleware* movem informações entre programas, ocultando a heterogeneidade do sistema. A heterogeneidade tanto de *hardware* quanto de *software* é um dos grandes problemas da computação distribuída. A criação de *middlewares* surge como solução para esse problema.

A *Interpercepção*, definiu um componente de *software* baseado em *middleware* para resolver o problema da criação de um ambiente virtual compartilhado para PC. A arquitetura e o protocolo de comunicação definido pela Interpercepção permitem que qualquer

usuário se conecte com um ambiente virtual em pelo menos três diferentes interfaces visualização.

Celulares e outros dispositivos possuem uma grande variedade de poder de processamento computacional. Essa variedade de recursos computacionais motivou a criação da *Interpercepção*. Se essa variedade de recursos é encontrada em outros dispositivos, isso nos levar a propor uma extensão da *Interpercepção* aplicada a diferentes dispositivos.

Diante do ambiente heterogêneo apresentado até o momento, levantamos a hipótese de que o conceito de Interpercepção pode ser aplicado a outros dispositivos. Dessa hipótese surge um novo escopo para interpercepção. Contudo, a ampliação desse escopo não altera o objetive inicial da *interpercepção* que é prover a interligação entre ambientes virtuais com interfaces de visualização distintas.

> **Hipótese 1.1**: A extensão da *Interpercepção* permitirá a execução de uma aplicação multiusuário *on-line* compartilhada entre múltiplos dispositivos e com suporte a diferentes interfaces visualização.

A Hipótese 1.1 propõem que a abordagem definida nesse trabalho permite a integração de usuários conectados pela *Internet*, usando aplicativos cliente diferentes. Esses clientes podem ser desenvolvidos em diferentes plataformas (o que emvolve o uso de diferentes linguagens de programação). A abordagem proposta permite a criação de ambientes virtuais para vários dispositivos e perfis de usuários.

Quando falamos em perfis de usuários surgem outras questões além da diversidade de interfaces de visualização. Os usuários possuem diferentes interesses e necessidades de acordo com o seu sexo, idade, nacionalidade, cultura, entre outros aspectos. Isso altera o modo como eles percebem e interagem com o ambiente. Um usuário cego, por exemplo, não seria capaz de acessar um ambiente virtual baseado apenas no *feedback* visial. Na abordagem da *Interpercepção* as formas de acesso são baseadas na diferença de interfaces visuais. Assim, um usuário cego estaria fora do escopo de uso da *Interpercepção*. Contudo, se adicionarmos ferramentas de síntese e reconhecimento de fala ao ambiente, permitiriamos o accesso de uma pessoa cega. Como para esse usuário o aspecto visual não importa, as ferramentas de síntese e reconhecimento de fala poderiam ser adicionadas a versão textual (1D) do ambiente.

O uso de ferramentas de síntese e reconhecimento de fala apresentam uma solução para atender às necessidades de um grupo de usuários. Através da adição de outras ferramentas é possível atender as necessidades de outros grupos de usuários. Todas esses ferramentas podem ser reunidas como serviços de um middleware. Esse middleware facilitará a criação de ambientes virtuis que atendem as necessidades de vários perfis de usuários. Assim, surge o principal objetivo dessa tese.

1.2 Objetivos

O principal objetivo dessa tese é alcançar o maior número de perfis de usuários dentro do escopo de um ambiente virtual. Para isso desenvolvemos uma abordagem baseada em middleware, chamada**GATE** (acrônimo de *Get All the Environments*). Essa abordagem

promove: a interoperabilidade entre os diferentes aplicativos clientes; a comunicação de todos os clientes com os servidores do sistema; a integração de clientes com interfaces de visualização distintas; a integração de usuários com diferentes perfis, através de adição de novas ferramentas que promovem o acesso desses usuários.

1.2.1 Objetivos Secundárias

Um conjunto de objetivos secundárias surge da realização desse trabalho. A concepção desta tese contempla as seguintes objetivos:

1. Desenvolvimento de uma arquitetura de *software* que estabelece o modelo da *Interpercepção* entre dispositivos. Essa arquitetura é fracamente acoplada o que permite que novos componentes de *software* sejam adicionados a mesma. A liberdade de adicionar componentes futuros possibilitará a integração futura de novos dispositivos ao modelo da *Interpercepção* entre dispositivos.
2. Desenvolvimento de uma infra-estrutura de servidores capaz de oferecer suporte ao tipo de aplicação que está sendo proposta neste projeto. São estes servidores que irão garantir que o sistema irá permitir a integração de vários usuários no mesmo ambiente. Estes servidores também deverão ser capazes de lidar com as requisições de usuários conectados por diferentes dispositivos.
3. Desenvolvimento de um protocolo de comunicação entre os diferentes dispositivos que poderão ser integrados através da *Interpercepção*. Este protocolo deverá garantir a sincronia do ambiente entre os diferentes dispositivos. Isso é possível graças a utilização de dados binários para construção e transmissão das mensagens do protocolo.
4. Desenvolvimento de algoritmos que permitam a conversão das informações trocadas entre os usuários de dispositivos diferentes. Este algoritmo deverá garantir que as informações visualizadas por algum usuário sejam devidamente adequadas a interface e ao dispositivo que ele está utilizando, afim de não gerar erros de incompatibilidade.
5. Uma reetruturação da *Interpercepção* para fornecer suporte a novos perfis de usuários. Esses novos perfis envolvem usuários com deficiências visuais e/ou auditivas, usuários que se comunicam por diferentes linguagens ou ainda que possuem diferentes recursos de *hardware* e/ou *software*.

1.3 Contribuições

A principal contribuição desta tese é ampliar o escopo da *Interpercepção*, propondo e implementando uma nova abordagem baseada em middleware denominada GATE. Essa abordagem agrega uma série de serviços de middleware para construção de ambientes virtuais baseados na *Internet*. Além disso, esse trabalho alcança as seguintes contribuições:

- O estabelecimento de um serviço de *middleware* que propicie a criação eficiente de aplicações *Interperceptivas* para múltiplos dispositivos. A *Interpercepção* demonstra sua viabilidade também em aplicativos para celular e outros dispositivos

de exibição (PDA, TV Digital etc). A partir da definição desta arquitetura, será possível estender a *Interpercepção* para criação de aplicações interativas da TV Digital, para dispositivos móveis e para quaisquer outros dispositivos.

- Melhoramos o paradigma da Interpercepcao introduzido anteriormente, ampliando seu escopo para novas possibilidades de percepção do ambiente, que vise permitir a integração de novos perfis de usuários aos ambientes criados com *Interpercepção*.
- Condensamos pesquisas dos últimos 10 anos do Natalnet em Realidade Virtual, Computação Gráfica e Redes, fazendo a engenharia do nosso modelo. Através dessa consolidação das nossas pesquisas trabalhos atuais estão sendo desenvolvidos sobre nosso paradigma e trabalhos futuros irão se beneficiar dele também.

1.4 Justificativa

Apesar de foco da *Interpercepção* ser os ambientes virtuais, qualquer tipo de aplicação que possua versões com interfaces distintas para visualização dos seus dados, pode utilizar a Interpercepção para promover a integração das versões. Como está sendo evidenciada, a Interpercepção lida com aplicações multiusuários e seu objetivo é agregar diferentes perfis de clientes num mesmo ambiente de interação compartilhado.

Com o uso da *Interpercepção* em aplicações para computador demonstramos que a existência de diferentes interfaces de visualização (e conseqüentemente de comunicação) garantem maior acessibilidade a um determinado sistema. Este aspecto permite que usuários com fatores limitantes possam também interagir com o sistema. Usuários com equipamentos de baixo desempenho que não conseguem participar de uma interface 3D, podem optar por interfaces menos exigentes, como a interface 2D, e usuários com deficiência visual, através das técnicas de síntese e reconhecimento de voz, podem utilizar o sistema através da interface textual. Assim a *Interpercepção* surge como uma alternativa para reunir a grande diversidade de espectadores da TV Digital. Ela também possibilitará a integração de usuários da versão móvel (celular) e do PC com o resto do sistema.

Como o foco da *Interpercepção* está no desenvolvimento de ambientes colaborativos, ela também ajudará a fortalecer seu desenvolvimento na TV Digital e no celular. Através de tais ambientes aplicações educativas, entretenimento [Benford et al. 2000, Nolan 2002], treinamento, vídeo conferências e outras aplicações poderão ser vistas por usuários de PC, TV celular de forma síncrona. Isto demonstra que a *Interpecepção* entre dispositivos é uma aplicação pervasiva [Szymanski & Yener 2005] e, portanto está atrelada a um dos conceitos fundamentais da computação nos próximos anos. A *Interpercepção* entre dispositivos também reflete um ideal de convergência tecnológico, uma das grandes preocupações atuais da computação [Tkachenko et al. 2004].

Quando um grupo de pessoas interage através de uma aplicação *Interperceptiva* elas quebram barreiras de espaço e também barreiras tecnológicas, já que a *Interpecepção* entre dispositivos como está sendo proposta neste trabalho permitirá o acesso através de diferentes plataformas. A possibilidade de um usuário cego utilizar uma aplicação *Interperceptiva*, através de uma ferramenta de síntese e reconhecimento de voz e com isso poder interagir com outro usuário qualquer, confere ao sistema um grau de acessibilidade pioneiro. Portanto a extensão da *Interpecepção* que está sendo proposta aqui, também

quebrará barreiras sociais. A acessibilidade no escopo da computação também é uma área de pesquisa em desenvolvimento e em franca expansão [Piccolo et al. 2007].

1.5 Metodologia

A metodologia e as estratégias de ação para o desenvolvimento desse trabalho se encontram elaboradas da seguinte maneira:

1. Até o presente momento a Interpercepção foi aplicada no escopo dos ambientes virtuais para PC. Agora que expandimos o conceito da Interpercepção para outros dispositivos, as metodologias e arquiteturas utilizadas para o desenvolvimento de ambientes virtuais deveriam ser revistas, pois podem não se adequar a este novo tipo cenário. Para tal um estudo foi realizado para se reavaliar as estratégias de construção de ambientes virtuais a fim de definir a que mais se adequava a este novo escopo da Interpecepção;
2. Uso da Interpercepção em ambientes virtuais multiusuários baseados na web prevê o uso da intra-estrutura da rede fixa (por cabo, como as redes ADSL) para a interligação dos usuários. Com a extensão da Interpercepção para celulares foi necessário estudar as arquiteturas de comunicação para dispositivos móveis e propor uma metodologia viável para a integração de celulares no escopo da Interpercepção. A viabilidade desta metodologia esta ligada diretamente ao custo do serviço de comunicação móvel para o usuário final de uma aplicação com Interpecepção. Se este custo for muito elevado à aplicação não será economicamente viável. Além disso, a taxa de transferência de dados em celular pode impedir a sincronização das informações entre os usuários do PC e os usuários do celular. Portanto é vital que seja proposto um conjunto de estratégias para superar estas dificuldades da comunicação por celular.
3. Como a transferência bidirecional de informação tem na interação seu elemento chave, aspectos e mecanismos de colaboração eficientes e flexíveis entre os participantes do ambiente de *Interpercepção* entre dispositivos serão fundamentais. Neste contexto, pretende-se estudar diversos mecanismos e estratégias de colaboração, de modo a se determinar quais se apresentam mais apropriadas a essa classe de sistemas. Para tal serão necessário os seguintes estudos:

   Estudar as arquiteturas e protocolos de comunicação para celular, PDA e outros dispositivos para definir uma melhor forma para construção de aplicações multiusuários para esses dispositivos.
4. Este trabalho está focado no desenvolvimento de aplicações multiusuários. Para desenvolver tais aplicações foi preciso realizar um estudo comparativo entre as várias abordagens destinadas à construção de sistemas distribuídos. Com base neste estudo foi possível desenvolver uma infra-estrutura de servidores capaz de oferecer suporte ao tipo de aplicação que está sendo proposta. Estes servidores serão responsáveis pela integração de vários usuários em tempo real no mesmo ambiente. Para realizar tal integração estes servidores também deverão ser capazes de lidar com as requisições de usuários conectados por diferentes dispositivos. Para realizar

a integração entre estes dispositivos diferentes foi necessário conhecer o protocolo de comunicação de cada um deles e propor um novo protocolo que permita a comunicação entre todos eles.

5. Para garantir a Interpercepção, cada dispositivo deverá ser capaz de exibir a informação de forma adequada a sua capacidade técnica. Assim, um dispositivo deve exibir as informações de acordo com a resolução permitida pelo aparelho. Contudo como as informações aqui citadas pertencem ao escopo de uma aplicação colaborativa, os usuários de todos os dispositivos compartilham essas informações. Para adequar estas informações para todos os dispositivos será necessário desenvolver algoritmos que permitam a conversão das informações trocadas entre os usuários de dispositivos diferentes. Para concepção destes algoritmos foi necessário:

   Estudar as diferentes formas de representação de mídias nestes dispositivos, a fim de determinar o formato de cada uma.

   Com base na definição destes formatos, definir um conjunto de meta-dados capaz de representar as informações trocadas entre os dispositivos.

   Estabelecer métricas para conversão de meta-dados em informação compatível com algum tipo de dispositivo;
6. Para a construção da arquitetura ideal para um sistema Interperceptivo, foram levados em consideração alguns aspectos tais como: as necessidades da aplicação, os tipos de usuários, e o grau de sofisticação dos dispositivos que podem ser integrados ao mesmo. Com base na análise destes aspectos foi possível propor alguns modelos arquiteturais que se encaixem neste de tipo de aplicação.
7. O estudo de padrões de projeto [Gamma 2001] previamente utilizados na construção de aplicações distribuídas e aplicações colaborativas serão vitais para propor a arquitetura do sistema aqui proposto. Serão avaliados exemplos precedentes da literatura do uso destes padrões para construção de aplicações do tipo que será abordado neste trabalho.
8. Levantar os requisitos necessários para os diferentes tipos de usuários de uma aplicação Interperceptiva para PC, celular, entre outros dispositivos. Além dos requisitos de software que uma aplicação deste tipo necessita atender, outros requisitos particulares a grupos de usuários também precisam ser considerados.
9. Aplicar a arquitetura proposta em um estudo de caso de maneira que as funcionalidades do software possam ser efetivamente testadas diante das necessidades dos usuários. Este estudo de caso serve para validar as funcionalidades propostas para uma aplicação interperceptiva.
10. O processo de análise dos requisitos vai depender diretamente do estudo de caso escolhido, principalmente por haver tipos de usuários distintos. Contudo, independentemente do estudo de caso, os usuários precisam ser ouvidos, de forma que possam passar suas reais necessidades relacionadas à visualização e análise de dados de uma aplicação Interperceptiva entre diferentes dispositivos. A partir dessa etapa, ficará fácil concluir que alguns usuários têm necessidades parecidas, e eles podem vir à forma um grupo específico situado dentro do objetivo do sistema Interperceptivo. A análise também facilita a utilização de método de agrupamento que vise formar grupos de usuários baseados no perfil dos usuários.

1.6 Organização geral deste trabalho

Após esta introdução, aqui finalizada, apresentamos no Capítulo 2 um embasamento teórico sobre os conceitos que são vitais para o entendimento e elaboração deste trabalho. No capítulo 3, são analisados alguns trabalhos que apresentam uma abordagem semelhante a nossa. A formalização do problema abordado neste trabalho, ou seja, o desenvolvimento de ambientes virtuais para execução em múltiplos dispositivos é tratada no Capítulo 4. A *interpercepção*, nossa abordagem para integrar os ambientes, é descrita e detalhada no Capítulo 5. No Capítulo 6 apresentamos os resultados dos testes e experimentos realizados para validar nossa proposta. Finalmente, no Capítulo 7, apresentamos nossa conclusão.

Capítulo 2

Embasamento teórico

No capítulo de introdução, o tema deste trabalho foi inserido no escopo dos ambientes virtuais multiusuários *on-line*. No momento que esses ambientes são executados *on-line*, eles se tornam um sistema distribuído também. O trabalho aqui apresentado ainda trata da criação de aplicações para ambientes de computação pervasiva

Para o entendimento deste trabalho, é essencial o conhecimento de uma série de conceitos, ligados às três áreas da computação aqui citadas: Realidade Virtual (focado no tema de Ambientes Virtuais), Sistemas Distribuídos e Computação Pervasiva. Existem ainda citações de alguns conceitos da área de Interação Humano-Computador. Os conceitos aqui apresentados estão agrupados em seções sobre cada uma das áreas da computação listadas neste parágrafo.

2.1 Realidade Virtual

A possibilidade de imersão em ambientes simulados vem motivando artistas, engenheiros e profissionais da mídia desde a invenção do cinema. A Realidade Virtual (também abreviado na literatura para RV) não é uma área de pesquisa tão recente. Apesar disso, ainda continua sendo tema de pesquisas atuais e está relacionada à tecnologia de ponta, buscando sempre interfaces interativas mais próximas aos sentidos humanos. A Realidade Virtual pode ser definida como uma interface avançada para aplicações computacionais, onde o usuário pode navegar e interagir, em um ambiente gerado por computador, em tempo real, usando dispositivos de simulação sensorial [Burdea & Coiffet 2003]. A simulação sensorial garante a imersão do usuário no ambiente simulado e seu conseqüente desligamento das sensações do mundo real.

A realidade virtual também pode usada para aprimorar as informações do mundo real. Ela pode ser usada no sentido inverso, ampliando a informação gerada no virtual através de dados colhidos no real. Isso nos leva a outros conceitos fundamentais da área: Realidade Aumentada e Virtualidade Aumentada. Quando um sistema de RV é capaz de gerar Realidade Aumentada e Virtualidade Aumentada, ele é tido como um sistema de Realidade Mista [Dubois et al. 2009].

2.1.1 Realidade Mista

Através da realidade aumentada (também abreviado na literatura para sigla RA) [Haller et al. 2007] imagens geradas por computador são sobrepostas à imagem do mundo real. Assim, podemos dizer que através de uma aplicação de RA é possível trazer para dentro do ambiente real, elementos oriundos do virtual. A realidade aumentada vem sendo amplamente empregada na educação atualmente [Billinghurst et al. 2002]. Através do emprego de suas técnicas é possível aprimorar a experiência visual de uma aula com visualização de modelos 3D sobre o assunto ministrado. A Figura 2.1 ilustra uma aplicação de RA.

Na Figura 2.1 temos na imagem da esquerda um modelo gerado a partir do reconhecimento de um padrão de imagem em um cartão. Uma câmera ligada a um computador capta esse padrão. Um aplicativo transforma esse padrão em um modelo 3D. Esse modelo é visualizado pelo usuário dos óculos. Na imagem da direita, informações adicionais são sobrepostas à visão do ambiente real. Os óculos usados nessa simulação são semitransparentes. Isso permite exibir de forma sobreposta os dados gerados por computador.

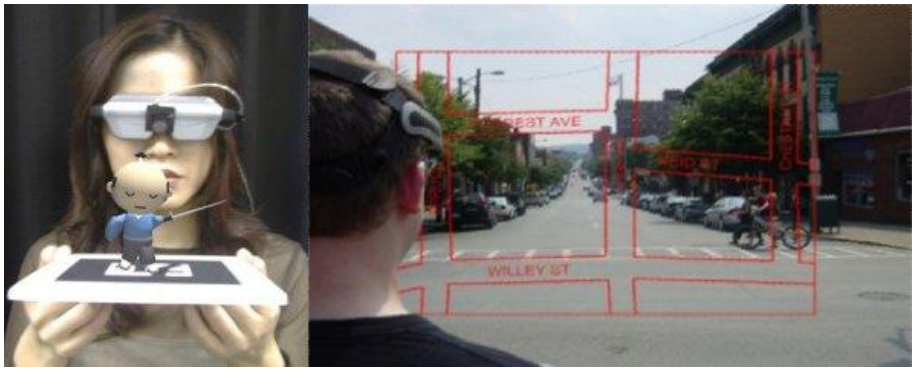

Figura 2.1: Exemplos de aplicações de Realidade Aumentada.

A virtualidade aumentada [Regenbrecht et al. 2004] faz o caminho inverso da realidade aumentada: leva elementos do mundo real para dentro de um ambiente virtual. Com as técnicas de virtualidade aumentada é possível, por exemplo, criar ambientes simulados onde são apresentados vídeos em tempo real. Esses vídeos são capturados do mundo real e exibidos dentro do ambiente simulado por computador.

Em um sistema que provê ferramentas de realidade aumentada e virtualidade aumentada é tido como um sistema de realidade mista [Ohta & Tamura 1999]. Um sistema de realidade mista fornece a integração total do ambiente real com o ambiente simulado. Esses ambientes simulados são referidos na literatura como ambientes virtuais.

2.1.2 Ambientes Virtuais

O conceito de ambientes virtuais foi introduzido por Ivan Sutherland [Sutherland 1965] em 1965 através de suas teorias sobre a inserção de pessoas em ambientes gerados por computador. Ele demonstrou suas teorias através da projeção de modelos tridimensionais em wireframes. Uma representação em wireframes utiliza vértices e arestas para

construção de objetos 3D. No momento do desenho na tela de um PC os vértices são substituídos por pontos com coordenadas 3D. As arestas são representadas por linhas entre os pontos, no momento do desenho.

Sutherland utilizou um par de telas montadas para projetar os objetos 3D. Esse par de telas foi montado em um tipo de capacete, chamados de *Head-Mounted Displays* ou HMD, apresentado na Figura 2.2. O HMD havia sido patenteado em 1960 pela Philco [Elllis 1991], mas foi o dispositivo desenvolvido por Sutherland em 1968 [Sutherland 1968] o primeiro a usar imagens geradas por computação gráfica.

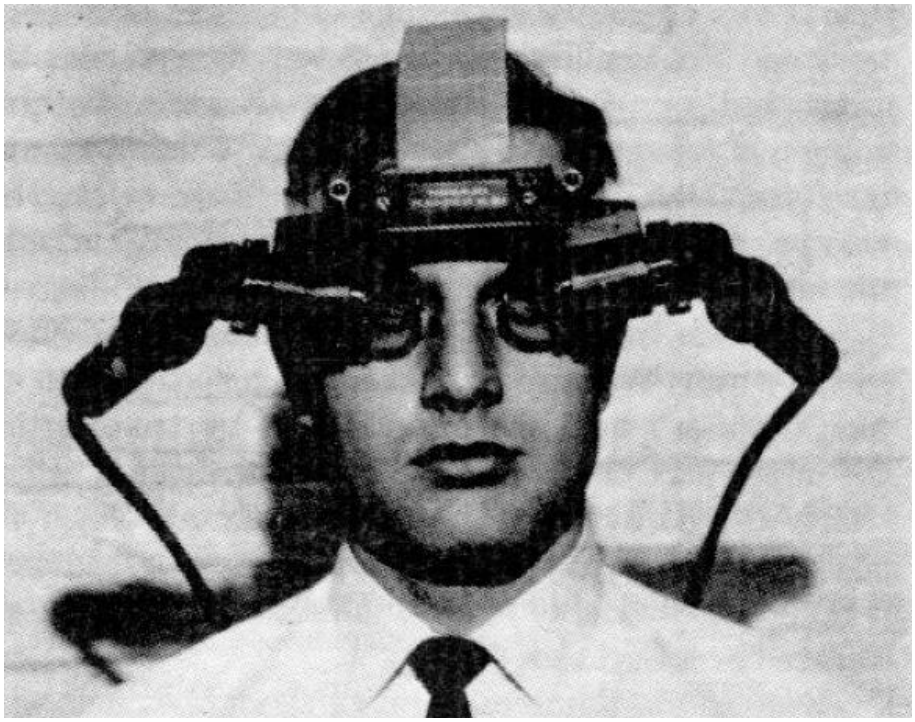

Figura 2.2: HMD desenvolvido por Sutherland.

Outra importante contribuição para consolidação do termo ambiente virtual foi dada por Steve Ellis [Ellis 1994]. Ele definiu o termo virtualização como sendo o processo pelo qual um espectador humano interpreta uma impressão sensorial que representa um objeto em um ambiente diferente daquele no qual a impressão se originou fisicamente. Ellis dividiu o processo de virtualização em três níveis:

1. **Espaço virtual**: espaço onde será projetada a imagem virtual;
2. **Imagem virtual**: percepção de profundidade dos objetos;
3. **Ambiente virtual**: o usuário torna-se parte do mundo virtual.

Unindo as teorias destes dois autores, podemos definir um Ambiente Virtual como sendo um ambiente sintético composto por uma modelagem gráfica gerada por computador e por um conjunto de elementos que estimulam os sentidos do usuário, simulando a sua presença neste ambiente. Considerados como mídia de comunicação, os ambiente virtuais têm ampla aplicação em educação, treinamento, programação de alto-nível, tele-operação

remota, exploração de superfícies planetárias, análise exploratória de dados, visualização científica e entretenimento. Atrelado ao conceito de ambiente virtual surgiram outros conceitos tais como comunidades virtuais, mundos virtuais, ambientes virtuais multiusuários, entre outros. A seguir apresentamos as definições sobre esses conceitos.

2.1.3 Mundo Virtual

Definimos acima o termo ambiente virtual, porém outro termo comum que encontramos em algumas pesquisas sobre o assunto é o termo mundo virtual (do inglês *virtual world*). Um mundo virtual pode ser entendido como um ambiente virtual dotado de uma representação espacial, regido por leis e grandezas físicas, principalmente o tempo. Esta definição demonstra que o conceito de ambiente virtual é mais abrangente que o conceito de mundo virtual, porém não é difícil encontrar estes dois termos sendo associados como se fossem sinônimos. A figura 2.3 mostra uma imagem da tela inicial do mundo virtual *on-line* chamado *There* [Makena Technologies 2009]. Ao acessar inicialmente um mundo virtual como o *There*, o usuário é convidado a criar uma representação virtual de si. Essa representação virtual e comumente chamada na literatura de avatar. Definimos o termo a avatar a seguir.

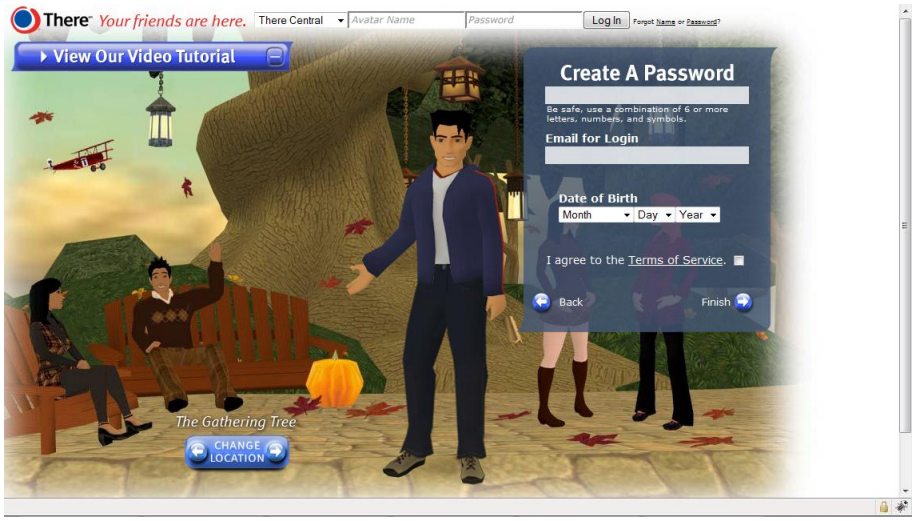

Figura 2.3: Interface do mundo virtual There.

2.1.4 Avatar

O termo avatar é originário da mitologia indiana. A palavra vem do *sânscrito Avatãra* que significa descida ou encarnação. É usada nos textos mitológicos indianos para se referir à materialização de um deus na forma humana, para poder caminhar sobre a terra. No contexto da realidade virtual, o termo avatar é usado para designar um ícone ou qualquer outra representação visual do usuário dentro de um ambiente virtual compartilhado [Damer 1997]. O avatar define o modo como os usuários do ambiente virtual

percebem uns aos outros. O termo foi usado pela primeira vez, com este significado, em 1985 no jogo Ultima IV. Foi popularizado por Neal Stephenson no seu romance *Snow Crash* (publicado em português com título Nevasca) [Stephenson 2007]. A figura 2.4 mostra exemplos de avatares.

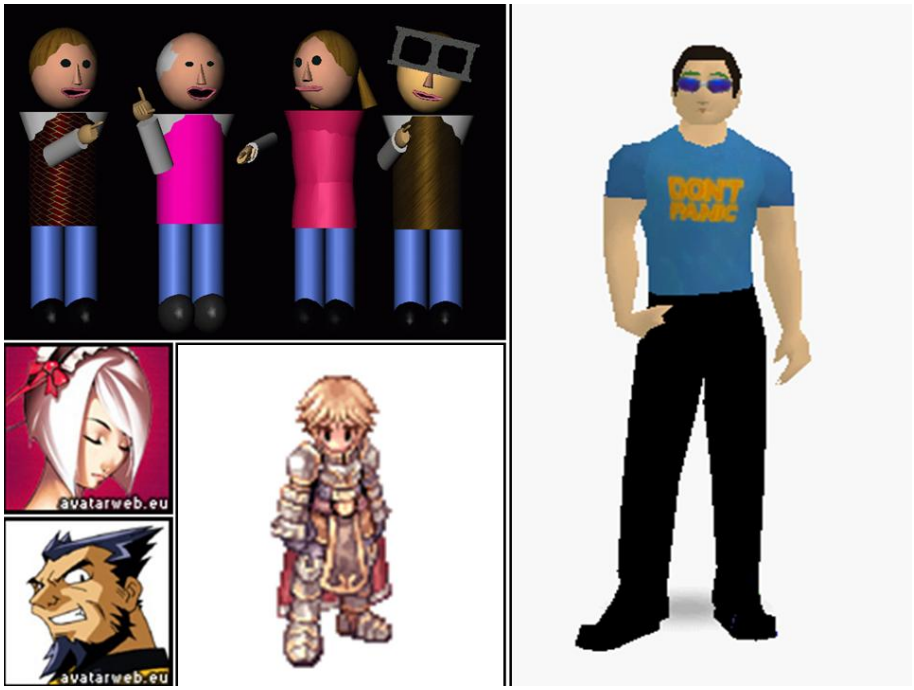

Figura 2.4: Exemplos de Avatares

2.1.5 Comunidades virtuais

O uso de avatares permite representar cada usuário que está conectado ao ambiente virtual. Através do seu avatar um usuário é capaz de se apresentar aos demais usuários. Da mesma forma, é possível perceber os avatares de todos os outros usuários conectados. Um uso comum de avatares na atualidade é a representação de usuários dentro de uma comunidade virtual.

Comunidades virtuais baseiam-se no conceito mais geral de comunidade. O termo virtual é acrescido para diferenciar sua natureza, ou seu campo de atuação, que, nesse caso, é o espaço virtual. Portanto, uma comunidade virtual nada mais é do que um grupo de pessoas interligadas por interesses, metas ou valores comuns, dentro de um espaço virtual.

A *Internet* tem sido atualmente este espaço virtual, onde surgem a cada dia novas comunidades virtuais, dentre as quais o Orkut [Google 2009] tem se destacado gradativamente no Brasil. A figura 2.5 mostra a tela inicial do Orkut.

Apesar de difundidas, as comunidades virtuais ainda não contemplam aquilo que geralmente esperamos de um mundo virtual: a modelagem topográfica que nos forneça uma

noção de espaço. Apesar de permitir uma representação visual dos seus vários integrantes, um serviço de comunidade virtual geralmente não oferece meios para interação dos usuários que estão conectados ao mesmo momento. Alguns serviços de criação de comunidades virtuais, como o Orkut, por exemplo, oferecem ferramentas de bate-papo embutidas. Se todos os meios citados fossem oferecidos aos usuários, as comunidades virtuais poderiam ser consideradas uma aplicação multiusuário. É neste ponto que introduzimos o conceito de ambiente virtual multiusuário.

Figura 2.5: Interface do Orkut

2.1.6 Ambientes virtuais multiusuário

Com base nos conceitos que já apresentamos até aqui, podemos definir ambientes virtuais multiusuários como sendo: ambientes caracterizados por suportar espaços sintéticos que são compartilhados por múltiplos usuários, remotamente localizados.

O primeiro mundo virtual multiusuário no qual as pessoas tinham uma representação gráfica 2D foi criado em 1985: O Habitat [Morningstar & Farmer 1991]. Desenvolvido por Randy Farmer e Chip Morningstar para a LucasFilm Games ele permitia aos usuários se comunicar e formar uma comunidade virtual. Apesar de ter sido proposto em 1985, o Habitat (versão beta) só foi liberada para testes em 1987. A Figura 2.6 ilustra a interface de usuário do Habitat. O Habitat constitui um ponto inicial na consolidação do conceito de ambientes virtuais multiusuário. Tornou-se popular na década de 90, adquirindo inclusive outras versões como Fujitsu Habitat e WorldsAway.

O acesso a estes ambientes através da *Internet* é importante, porque permite que uma ampla gama de aplicações, de entretenimento à educação e medicina, seja oferecida, através de uma interface bastante difundida como os navegadores da *web*. A figura 2.7 apresenta uma imagem de demonstração do River City [Dieterle & Clarke 2007], um ambiente virtual multiusuário para *web* desenvolvido por pesquisadores na universidade de Havard. O intuito do projeto River City é servir como uma plataforma auxílio ao ensino de ciências e tecnologia para alunos do ensino médio.

Figura 2.6: Interface do Habitat

Outra vertente das pesquisas sobre ambientes virtuais que tem crescido bastante é a dos ambientes virtuais de grande escala (massivo) [Greenhalgh 1999]. A adição do termo larga escala apenas amplia o conceito de ambientes virtuais multiusuário para ambientes capazes de suportar uma quantidade muito grande de usuários. A construção de ambientes de larga escala tem sido impulsionada principalmente pela indústria de jogos, após o surgimento dos MMOGs (*Massive Multiplayer Online Game*). Os MMOGs nada mais são do que jogos multiusuários capazes de suportar milhares de jogadores conectados ao seu mundo virtual através da *Internet*, ou até mesmo através de uma LAN (*Local Area Network*).

A definição de ambientes virtuais multiusuário encerra a apresentação dos conceitos sobre ambientes virtuais que são relevantes para a compreensão desse trabalho. Através do uso de ambientes virtuais multiusuários é possivel validar a nossa abordagam para interligação de ambientes com visualização distinta.

Para conexão dos usuários do ambiente é preciso estabelecer uma infra-estrutura de comunicação entre esses usuários. Isso nos leva agora à definição de uma estrutura que permita a integração dos usuários remotamente localizados. Por isso entramos agora nos conceitos de Sistemas Distribuídos.

2.2 Sistemas Distribuídos

Segundo Tanenbaum & Van Steen [Tanenbaum & van Steen 2001], um sistema distribuído é "uma coleção de computadores independentes que se apresentam ao usuário como um sistema único e consistente". Num sistema como esse, o poder computacional de todas as máquinas envolvidas é somado. Este poder é então direcionado para a realização de uma tarefa colaborativa. A execução desta tarefa ocorre de forma transparente para o usuário, uma vez que para o ele todo o processamento é aparentemente feito por uma única máquina. Mesmo que o usuário tenha noção de que o processamento é feito por várias máquinas, ele ainda não seria realmente capaz de definir que parte da tarefa está

Figura 2.7: Interface do River City

sendo realizada por qual máquina, o que ainda configuraria o sistema como transparente.

Um sistema distribuído não se configura apenas por operar em várias máquinas interligadas, pois o poder computacional gerado por essa ligação também pode ser alcançado com a utilização de vários processadores trabalhando em paralelo num mesmo computador. Quando um sistema distribuído é construído desta forma, ele está seguindo uma arquitetura multiprocessadores, sendo a memória compartilhada por todos os processadores.

Se existe uma memória dedicada para cada processador ele é um sistema distribuído de multi-computadores, mesmo que todos os processadores e memórias estejam numa mesma máquina ou máquinas diferentes.

A principal diferença entre sistemas com um processador e sistemas distribuídos está na comunicação entre os processos, também definida como comunicação inter-processos. Em sistemas com um processador apenas, essa comunicação pode ser realizada através de uma estrutura de memória compartilhada. Nos sistemas distribuídos a comunicação inter-processos ocorre através de modelos (ou arquiteturas) para comunicação distribuída. O modelo de comunicação distribuída utilizado nesse trabalho é a Arquitetura Cliente-Servidor, que definimos a seguir.

2.2.1 Arquitetura Cliente-Sevidor

Essa arquitetura define um modelo de comunicação estabelecido por dois componentes básicos: o cliente e o servidor. O cliente é um processo que atua na requisição de informações. Essa requisição é feita ao servidor. O servidor tem a tarefa de atender às requisições do cliente. Outro modo de definir o papel de cada componente da arquitetura é tratar o servidor como um prestador de serviços, o que evidência bem o seu nome. Já o cliente é alguém que busca algum tipo de serviço oferecido pelo servidor.

A Figura 2.8 mostra um sistema seguindo a arquitetura cliente-servidor. Nesta figura temos apenas um cliente e um servidor, contudo é comum a existência de um conjunto de clientes buscando as informações no servidor. Nesta figura o cliente e o servidor utilizam um protocolo simples de requisição (enviada pelo cliente) e Resposta (enviada pelo servidor).

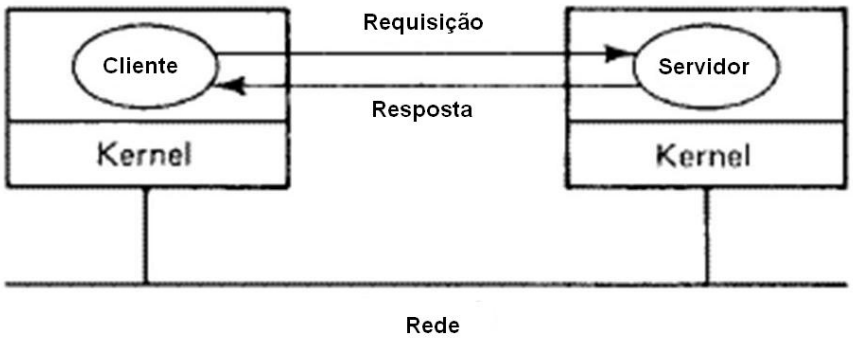

Figura 2.8: Arquitetura Cliente-Servidor

Exemplos de serviços que utilizam tal arquitetura são facilmente encontrados, como por exemplo, alguns serviços de Bate-Papo, alguns Jogos *On-line*, HTTP, DNS, entre outros. Nesses serviços há mais de um cliente e possivelmente mais de um servidor também, obrigando a uma estruturação do sistema no que diz respeito a como os clientes e servidores se relacionam.

O modelo cliente-servidor é usado na nossa abordagem, pois a mesma utiliza o arcabouço H-N2N. Esse arcabouço estabelece uma infra-estrutura para crição de ambientes virtuais multiusuários baseados numa cadeia hierárquica de servidores. Ele será descrito em mais detalhes no capítulo 4.

Existem outros modelos de comunicação distribuída além do cliente-servidor, como a *Peer-to-Peer* [Schollmeier 2002]. Ao invés de explorarmos outros modelos de comunicação distribuída, tratamos doravante do conceito de comunicação em grupos.

2.2.2 Comunicação em Grupos

Segundo Tanenbaum [Tanenbaum & van Steen 2001], um grupo é "uma coleção de processos que agem juntos em algum sistema ou de alguma forma específica". Em um sistema distribuído, o número de processos envolvidos numa comunicação pode variar

desde uma simples troca de dados entre dois processos até o compartilhamento de dados entre milhares de processos. Portanto, é necessária a existência de uma estrutura de coordenação entre esses vários processos. Para realizar tal coordenação é vital que seja desenvolvido um modelo para comunicação em grupo, o que permite desta forma a criação de grupos de processos que cooperem para fornecer um serviço. Basicamente, a implementação dessa comunicação pode ser feita através de três modelos: *unicast*, *multicast* e *broadcast*.

No modelo *unicast* a transmissão é realizada ponto a ponto. Nesse modelo o processo deve enviar a mensagem para cada um dos destinatários. No modelo *muticast* cada mensagem é enviada uma só vez e somente para os integrantes do um determinado grupo. Por último, no modelo *broadcast* a mensagem é enviada para todos os processos, e depois são filtradas pelos processos interessados naquela mensagem.

A nossa abordagem utliza comunicação nesses três modelos, descritos acima. Essa característica é também oriunda do uso do arcabouço HN2N. Ele permite que os clientes se comuniquem com um outro cliente através de comunicação *unicast*. O arcabouço organiza clientes em grupos, como por exemplo na representação de uma sala, de um ambiente virtual. Ele também permite criar mensagens que são enviadas a todos os usuários através de *broadcast*, alcançando todos os clientes de uma só vez.

2.2.3 Acoplamento

Uma métrica importante no desenvolvimento de um sistema distribuído é o acoplamento entre os módulos [Pressman 2006]. O acoplamento define quão interligado está um conjunto de módulos de um sistema. Quanto mais interligados mais dependentes eles estarão entre si. Conseqüentemente, a modificação de um módulo afetará os módulos dependentes. Isso é um problema sério para manutenção do *software* [Grubb & Takang 2003] do sistema.

O acoplamento define duas medidas básicas: forte acoplamento e fraco acoplamento. Um sistema fortemente acoplado é um sistema onde seus módulos são muito dependentes entre si. Já um sistema fracamente acoplado, os módulos são mais independentes. O ideal é construir sistemas fracamente acoplados, pois esses são mais fáceis de manter e expandir. Pela eficiência do fraco acoplamento, nós optamos por usar o mesmo em nossa abordagem.

2.2.4 Sistemas baseados em RPC

O RPC (*Remote Procedure Call*) [Laurent et al. 2001] é um modelo importante em computação distribuída. Proposto por [Birrell & Nelson 1984], com o objetivo de permitir a interação entre dois processos que executam em máquinas distintas, como se estivessem executando localmente. A Figura 2.9 apresenta uma arquitetura para comunicação usando RPC.

A Figura 2.9 mostra a arquitetura RPC como um sendo um modelo baseado na arquitetura cliente-servidor. Ambos os lados apresentam uma organização clara em três componentes distintas: Aplicação, *Stub*, e Biblioteca. No componente de Aplicação está

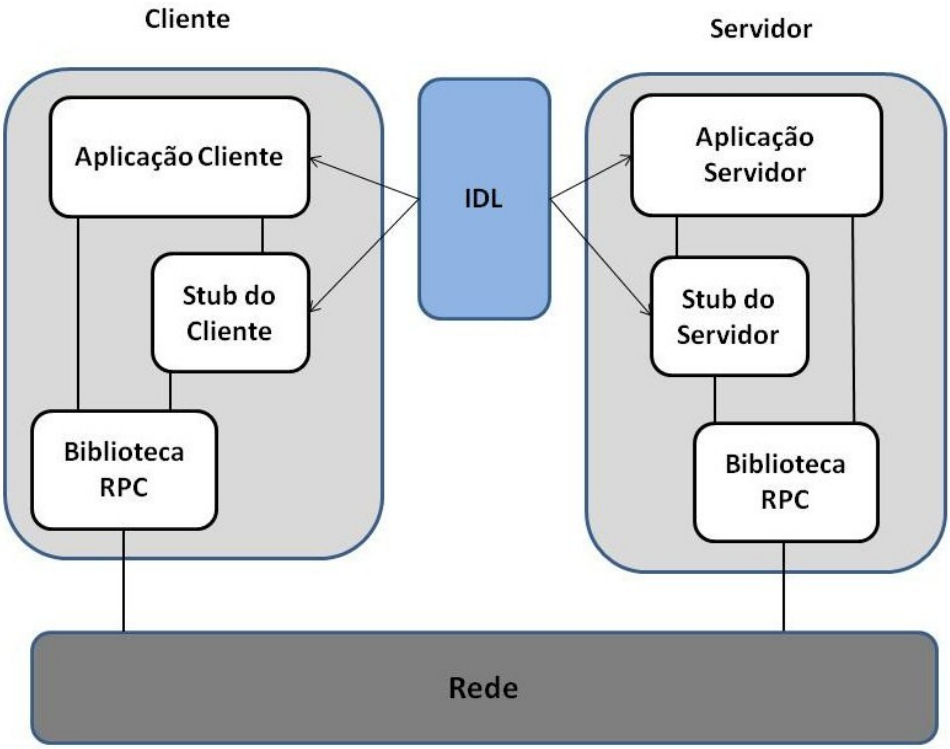

Figura 2.9: Exemplo de comunicação RPC

o *software* principal que deseja estabelecer comunicação com outro *software* qualquer através da rede. De cada lado é estabelecido um *Stub*, que atuam como tradutores da informação passada de um lado para o outro do sistema. Através dos *Stubs* fica estabelecido um protocolo de comunicação comum entre os aplicativos servidor e cliente. Como é possível existir diferentes protocolos de comunicação na camada de transporte da rede, a Biblioteca RPC é colocada como interface com a rede.

Os *Stubs* são definidos com base em uma IDL (*Interface Description Language* ou Linguagem de Descrição de Intreface). Dessa forma os *Stubs* seguem o mesmo padrão de criação e, portanto são capazes de traduzir a informação de outro *Stub*. Dentro dos *Stubs* são definidas funções que serão usadas na camada de aplicação. É através dessas funções que o *software* da aplicação realiza a chamada remota. De certa forma, essas funções também estabelecem um protocolo de comunicação abstrato entre os *Stubs*.

O RPC é baseado no modelo de interação síncrona. Isso significa que ao realizar uma chamada, um determinado processo A entra em estado de espera, até que receba a resposta do processo chamado. A interação síncrona é o que garante que a chamada remota funcione como uma chamada local (de natureza síncrona).

O protocolo de comunicação abstrato entre os *Stubs* deve ser definido cuidadosamente, do contrário surgirão falhas de comunicação. Ferramentas podem ser usadas para geração automática de *Stubs*. O uso dessas ferramentas facilita o processo de criação dos mesmos, à medida que torna o protocolo entre eles compatível.

O RPC é utilizado em muitas plataformas de desenvolvimento distribuído, como CORBA [Lang & Schreiner 2002], RMI [Grosso 2002], DCOM [Peiris et al. 2007], entre

outros. Na seqüência discutiremos algumas dessas implementações.

2.2.5 RMI

O RMI é uma API da linguagem Java que permite a execução de chamadas remotas ao estilo RPC. A criação do RMI foi motivada pelo fortalecimento das linguagens orientadas a objetos.

O RMI permite que um objeto de uma máquina virtual Java invoque métodos de outro objeto localizado remotamente (em outra máquina). A invocação é feita de forma transparente. Além disso, o RMI permite que o programador se abstraia de detalhes como endereçamento e comunicação.

Uma vez que o RMI é baseado no modelo RPC, ele prevê uma organização baseada na arquitetura cliente-servidor. Contudo, ao invés de usar uma IDL para definição de um protocolo de comunicação, o RMI utiliza um registrador de nomes (RMI *Registry*). Seria impraticável se para cada invocação de método remoto fosse necessário incluir o par <*máquina,porta*> para identificar onde se encontra o objeto que contém o método. Por isso o RMI oferece um serviço de registro de nomes. Esse serviço oferece informações sobre a localização de objetos remotos. Ele executa em um endereço bem conhecido, tanto pelo cliente quanto pelo servidor.

No funcionamento de um sistema construído com RMI, o servidor define objetos que o cliente pode usar remotamente. Os clientes podem invocar métodos nesse objeto remoto como se ele estivesse executando localmente. O RMI esconde o mecanismo subjacente de transporte, via rede, de argumentos dos métodos e dos valores de retorno.

Por ser baseado na linguagem Java, o RMI atrela o uso dessa linguagem na criação do sistema. O uso de RMI também pressupõe o uso do paradigma OO na construção do sistema. Isso gera limitação no desenvolvimento. Essas características também não são úteis no tratamento da heterogeneidade do sistema, que pode possuir clientes escritos em diferentes linguagens. Um determinado dispositivo que integra o sistema pode estar habilitado para apenas um tipo de linguagem. Por isso, é essencial que o sistema forneça suporte para comunicação entre diferentes linguagens de programação. Corba surge como uma solução para este caso.

2.2.6 CORBA

CORBA (acrônimo de *Common Object Request Broker Architecture*) é uma arquitetura para estabelecer e simplificar a troca de dados entre sistemas distribuídos heterogêneos. Sua principal vantagem é fornecer suporte a diversas linguagens, possibilitando a comunicação entre módulos escritos em linguagens distintas. Foi definido pelo OMG (*Object Management Group*) em 1991. A arquitetura CORBA é mostrada na Figura 2.10.

A arquitetura CORBA define o ORB (*Object Request Broker*). Ele atua como um módulo intermediário entre o cliente e um objeto remoto. Na definição de CORBA, um objeto é um componente de *software* qualquer que é parte integrante do sistema.

O ORB é responsável em aceitar a requisição do cliente e enviá-la para o objeto competente. A requisição do cliente ocorre através da invocação de métodos do objeto remoto.

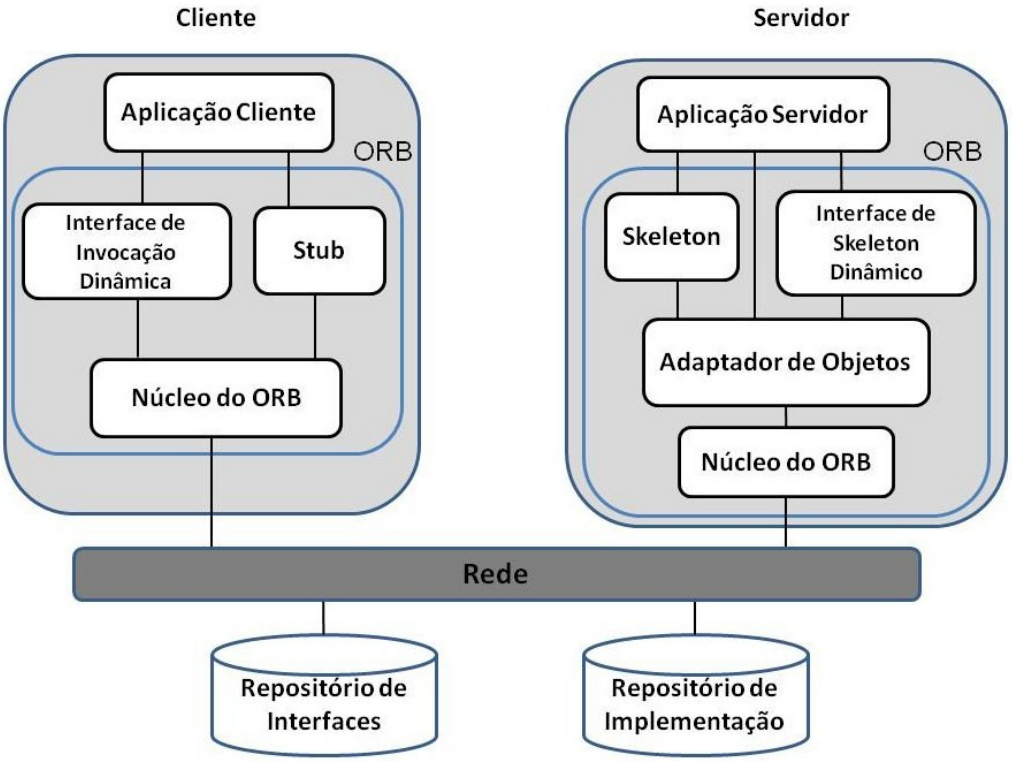

Figura 2.10: Arquitetura do CORBA

Quando a resposta estiver disponível, o ORB se encarrega de entregá-la para o cliente. Como apenas o ORB conhece a localização do objeto, a comunicação proporcionada pelo mesmo é tida como transparente ao cliente.

Uma vez que CORBA é uma implementação RPC, ela também utiliza uma IDL para criação de mediadores entre a aplicação cliente e servidor. A IDL de CORBA é baseada na linguagem C++. Do lado cliente a IDL é usada para gerar o *Stub*. O *Stub* é gerado com um compilador IDL, na linguagem de programação do próprio cliente. Do lado servidor existe o equivalente, chamado de *Skeleton*, que é gerado na linguagem do servidor. Num sistema com vários servidores, o *Skeleton* ainda realiza a tarefa de despachar uma requisição de um cliente ao servidor apropriado.

O Adaptador de objeto interliga os objetos CORBA e as classes do servidor. Ele cria as referências a objetos remotos para os objetos CORBA. Ele despacha cada invocação de método através de um *Skeleton* para o servidor apropriado. Ele também é responsável por ativar os objetos, se necessário. Versões recentes do CORBA o Adaptador ainda permite as que aplicações clientes e os servidores sejam executados com ORB's produzidos por desenvolvedores diferentes.

A Interface de Invocação Dinâmica permite que os clientes façam invocações dinâmicas a objetos CORBA desconhecidos. O cliente obtém informações necessárias sobre um objeto CORBA a partir de um Repositório de Interfaces, e utiliza essa informação para construir uma invocação e enviá-la ao servidor. O Repositório de Interfaces provê informações sobre interfaces IDL registradas (como métodos, argumentos, exceções). Um cli-

ente sem *Stub* para um determinado objeto pode obter informações necessárias (métodos e argumentos) para invocações dinâmicas. Nem todos os ORBs provêm um repositório de interfaces

A Interface de *Skeleton* dinâmico permite que um objeto CORBA aceite invocações em uma interface sem *Skeleton*. A interface não possui *Skeleton*, pois não era conhecida em tempo de compilação. O *Skeleton* dinâmico ao receber a invocação inspeciona o conteúdo da mesma. Ele descobre o objeto de destino e o método para ser invocado. Por fim ele invoca o objeto de destino.

Apesar de ser uma tecnologia robusta e bem difundida, CORBA apresenta alguns problemas. A transparência de localização dos objetos funciona de uma forma tão rígida que faz com que a invocação de uma função (método) local seja tratada da mesma forma que uma invocação a uma função remota. Isso diminui o desempenho do sistema. O CORBA foi projetado para atender a um conjunto de requisitos propostos por seus idealizadores. Não foi feita uma análise da coerência desses requisitos. Isso fez com que ao longo do tempo a especificação se tornasse muito complexa, de implementação custosa e muitas vezes ambígua.

2.2.7 Problemas inerentes ao RPC

As abordagens baseadas em RPC padecem de um problema crucial: comunicação síncrona. A sincronia restringe o funcionamento dos sistemas baseados em RPC, pois implica na disposição constante do servidor e do cliente. Isso reduz drasticamente o desempenho do mesmo.

Outro problema comum aos sistemas baseado em RPC é o forte acoplamento dos seus componentes. A comunicação síncrona e o próprio uso dos *Stubs* contribuem para tornar um sistema baseado em RPC fortemente acoplado. O advento da *Internet* favoreceu o crescimento de sistemas distribuídos. Deste fator surge também a necessidade da garantir que o sistema possa crescer em larga escala. O crescimento em larga escala é dificultado pelo uso de arquiteturas fortemente acopladas.

Sistemas modernos operam em ambientes complexos com múltiplas linguagens de programação, plataformas de *hardware*, sistemas operacionais. Desse panorama surgem exigências de flexibilidade de distribuição, confiabilidade, alto desempenho e segurança enquanto é mantida uma alta qualidade de serviço. Em tais ambientes, a abordagem direta de mecanismos do tipo RPC falhará rapidamente frente aos desafios atuais.

Com base nos problemas inerentes ao RPC, foi proposta a abordagem baseada em *middleware* para criação de sistemas heterogêneos.

2.3 *Middlewares*

O contexto computacional onde os sistemas distribuídos estão inseridos é heterogêneo. Existe uma grande diversidade de *hardwares* e SO (Sistema Operacional). A grande dificuldade de trabalhar neste ambiente heterogêneo é estabelecer a comunicação entre essas plataformas distintas. Esse quadro fez surgir uma camada de *software* entre o *hardware* e a API (*Aplication Programing Interface*), que estabelece uma ponte para facilitar

a comunicação. Esta camada é chamada de *middleware* [Vinoski 2004, Bernstein 1996]. A Figura 2.11 ilustra uma arquitetura de um sistema distribuído com *middleware*.

Como mostra a Figura 2.11, o *middleware* atua como uma ponte entra as aplicações e as diferentes plataformas. A arquitetura dessa figura ainda prevê a existência de interfaces definidas entres o *middleware* e as aplicações, através do uso de API (*Application Programing Interface*). De forma semelhante são estabelecidas interfaces de comunicação entre o *middleware* e cada uma das plataformas em separado.

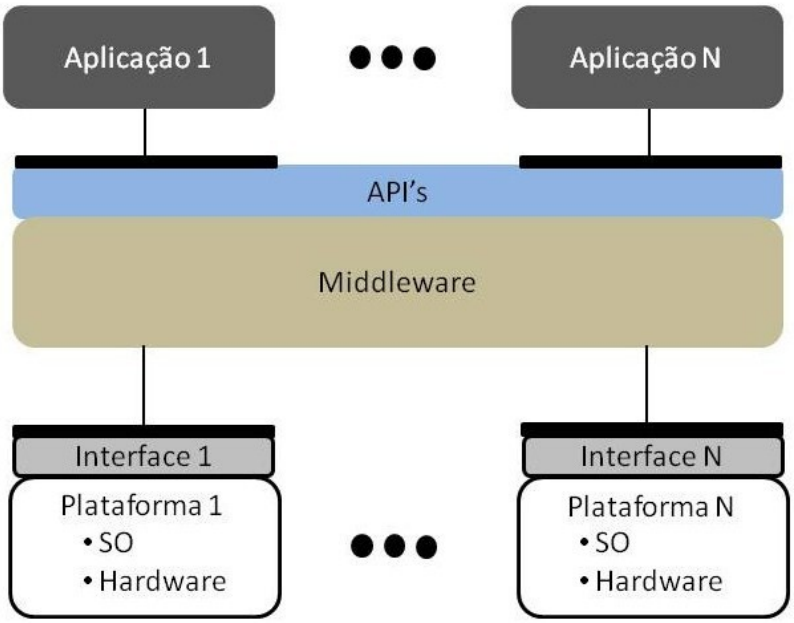

Figura 2.11: Arquitetura de *middleware*

Middlewares são camadas de *software* que não constituem diretamente aplicações, mas que facilitam o uso de ambientes ricos em tecnologia da informação. Eles fazem a mediação entre outros softwares, movendo informações entre programas e ocultando do desenvolvedor as diferenças de protocolos de comunicação, plataformas e dependências do sistema operacional. Seu objetivo é mascarar a heterogeneidade e fornecer um modelo de programação mais produtivo para os desenvolvedores de aplicativos.

Um *middleware* está geralmente localizado numa camada de *software* acima das plataformas dedicadas (*hardware*, SO, etc.) e logo abaixo das APIs que são usadas para desenvolver as aplicações. Como existe uma grande heterogeneidade de plataformas, o *middleware* servirá como um conversor entre uma aplicação e os diferentes tipos de plataformas nas quais esta aplicação poderá executar.

O *middleware* se trata de um sistema de alto nível. Por isso, são difíceis de serem definidos de forma técnica e precisa. Contudo, os *middlewares* possuem propriedades básicas que tornam claro como os mesmos se comportam: os *middlewares* são genéricos, executam em várias plataformas, eles são distribuídos, eles suportam interfaces e protocolos básicos.

2.3.1 Sistemas baseados em Middleware

Diante dos problemas de sincronia acarretados pelo modelo RPC, algumas abordagens, como CORBA, receberam atualizações para permitir comunicação assíncrona. Contudo, os críticos do CORBA apontam sua grande complexidade como uma barreira para o desenvolvimento. Abordagens como o RMI tentaram solucionar a complexidade do CORBA oferecendo uma implementação dependente de plataforma.

O interesse dos desenvolvedores de aplicações é que exista uma interface de programação única e independente de plataforma. Além disso, essa interface de programação deve ser capaz de eliminar a complexidade das chamadas de funções específicas de um determinado sistema operacional. Essa interface de programação de alto nível deveria abstrair as complexidades de redes e de protocolo, permitindo ao desenvolvedor focar somente na aplicação. Essa abordagem criaria um ambiente produtivo e onde o desenvolver tem uma única aplicação que executaria em várias plataformas. A resposta a essa necessidade é o *middleware*.

Segundo [Vinoski 2004], os *middleware* oferecem serviços de alto nível, independentes de plataforma e reusáveis, para sistemas distribuídos. Com isso, os *middleware* criam uma "homogeneidade virtual"no sistema, reduzindo a complexidade de criação de aplicações. Esses serviços oferecidos pelo *middleware* baseiam-se em dois procedimentos básicos:

1. Criação de padrões de interface de programação, que tornaria mais fácil a portabilidade de aplicações entre dispositivos e servidores;
2. Padronização de protocolos de comunicação.

2.3.2 Middleware Transacional

A origem dos *middleware* reside no desenvolvimento de Sistemas de Processamento de Transações (do inglês *Transaction Processing System* ou ainda TPS) [Bernstein 1996]. Nesse tipo de sistema um conjunto de pequenas operações é tratado como se fosse uma única operação. Esse mecanismo facilita a escrita de aplicações transacionais escaláveis e confiáveis.

A divisão de uma transação em pequenas operações é chamada de atomicidade da transação. O termo se refere ao fato das operações serem atômicas (indivisíveis em operações menores). As operações atômicas por terem um grau de complexidade muito baixo são mais confiáveis. Elas são indicadas para sistemas de alta confiabilidade, como por exemplo, sistemas bancários.

Sistemas desse tipo também são conhecidos por sua capacidade de recuperação de falhas. Em caso de falha, uma operação de refazer é acionada. A transação só entra em estado de completada caso não ocorra falhas na execução. A resposta do sistema também é rápida, graças a atomicidade das operações.

O principal problema dessa abordagem é sua inflexibilidade. Para garantir as qualidades desse tipo de sistema, é preciso garantir que todas as transações sejam executadas da mesma forma. A rigidez desse tipo de sistema originou a evolução dos sistemas de *middleware* para uso em aplicações mais flexíveis. Dessa evolução surgiram os *middlewares*

orientados a mensagem.

2.3.3 *Middlewares* Orientados a Mensagens

O Middleware Orientado a Mensagens (ou MOM) [Curry 2004] surge como um mecanismo alternativo ao RPC visando atender a demanda dos sistemas distribuídos. O MOM provê um método simples de comunicação entre diferentes entidades de *software*. Ele pode ser definido como uma infra-estrutura de *middleware* que provê capacidade de trocas de mensagens, e provê comunicação distribuída baseada no modelo de interação assíncrona.

Clientes de um sistema baseado em MOM podem enviar e receber mensagens de e para outros clientes através de uma camada de *middleware* que atua como intermediário. Essa mesma camada pode servir de intermédio entre os clientes e os servidores. A Figura 2.12 ilustra uma arquitetura baseada em MOM.

Pela arquitetura MOM, os cliente e servidores não precisam saber a localização uns dos outros, isso torna o sistema transparente. A falta de ligações diretas entre as aplicações do sistema torna o mesmo fracamente acoplado. O fraco acoplamento é também garantido pelo modelo de comunicação assíncrono. Por isso, a qualquer momento um dos componentes do sistema pode ser removido sem danificar o funcionamento do sistema. Novos componentes podem ser adicionados também.

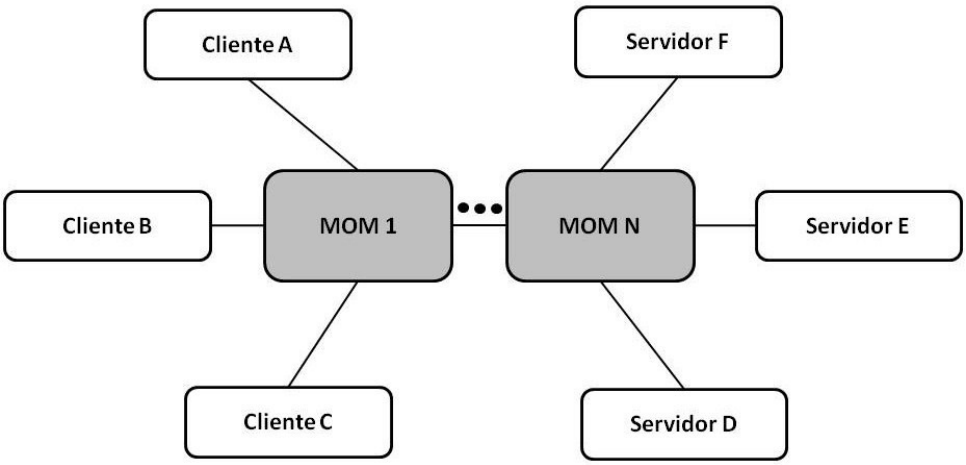

Figura 2.12: Exemplo de arquitetura de MOM

Para garantir o funcionamento assíncrono, o MOM estabelece uma fila de mensagens. À medida que as mensagens chegam ao mesmo, são enfileiradas. As mensagens são então tratadas de acordo com sua posição na fila.

As especificações mais recentes do CORBA permitem o uso de um modelo baseado em mensagens. Assim, nessa especificação a comunicação assíncrona é permitida mesmo usando CORBA. Contudo, isso não chega a reduzir a complexidade inerente ao uso de CORBA. Por isso, em nosso trabalho optamos por usar MOM, ao invés de RPC e CORBA.

2.3.4 Serviços de Middleware

Ao apresentar as diferentes abordagens baseadas em *middleware* é possível observar que elas apresentam propriedades necessárias a cada abordagem. Na abordagem orientada a mensagens, é garantida a comunicação assíncrona, rivalizando com o modelo RPC. Contudo, o CORBA, apesar de baseado em RPC permite a criação aplicações orientada a mensagens. Desse modo, podemos concluir que CORBA é também uma abordagem baseada em *middleware*. Na verdade o próprio RPC também pode ser visto com uma abordagem baseada em *middleware*, para aplicações síncronas.

A grande diferença entre cada abordagem esta nas propriedades garantidas, ou colocando de outra forma, nos serviços fornecidos. No início da discussão sobre sistemas baseados em *middleware*, foi citado que: "os sistemas de *middleware* oferecem serviços de alto nível, independentes de plataforma e reusáveis". Ao apresentar cada abordagem e listar suas propriedades, torna-se mais claro a afirmação acima. Por isso, podemos concluir que as propriedades de um *middleware* representam serviços que são fornecidos às camadas do sistema existente entre o mesmo.

2.4 Computação Pervasiva

A computação ubíqua (Ubicomp) foi introduzida em 1988 por Mark Weiser [Weiser 1991*b*] significando computação em todo lugar, o tempo todo. Já o termo computação pervasiva refere-se à tecnologia computacional embutida no nosso dia-a-dia. A pervasividade trata da capacidade computacional colocada nos dispositivos de maneira a produzir ambientes ubíquos.

Embora os termos ubíquo e pervasivo tenham significados diferentes, eles estão intimamente relacionados e tem sido usados de forma indistinta na literatura recente, assim como neste trabalho. O que caracteriza a computação ubíqua/pervasiva é a integração e interação entre vários dispositivos diferentes e aplicações. Isso é possível através do uso de *middlewares*.

As principais propriedades de um sistema pervasivo são a sensibilidade ao contexto, mobilidade e integração [Niemelä & Vaskivuo 2004]. A sensibilidade ao contexto define colaboração entre o sistema e as aplicações. O sistema monitora as aplicações e notifica a necessidade de mudança e então a aplicação decide como mudar.

A mobilidade é dividida em efetiva, virtual e física. A mobilidade efetiva diz respeito a alocação de recursos de forma dinâmica para os nós do sistema onde são necessários. A virtual diz respeito a localização e adaptação de redes existentes no ambiente de execução e a mobilidade física representa os próprios dispositivos moveis e a infra-estrutura de *hardware*.

A integração diz respeito à arquitetura que promove a interligação dos dispositivos. A definição dessa arquitetura descreve uma solução para os problemas de interoperabilidade entre os componentes do sistema que estão interagindo.

2.4.1 Middleware para computação Pervasiva

No capítulo 2 foi apresentado o conceito de computação pervasiva. Algumas qualidades inerentes a computação pervasiva foram descritas. Naquela mesmo capítulo foi citado o uso de *middleware* para criação de sistemas pervasivos. Um *middleware* projetado para esse tipo de sistema deve ser capaz de garantir um conjunto de propriedades básicas: sensibilidade ao contexto, mobilidade e integração.

Além das propriedades básicas podem ser adicionadas ao *middleware* de um sistema pervasivo, outras propriedades como: interoperabilidade, escalabilidade, adaptabilidade, robustez, confiabilidade [Niemelä & Vaskivuo 2004], gerenciamento de contexto, interação transparente com o usuário e invisibilidade [da Costa et al. 2008].

Interoperabilidade e escalabilidade (crescimento em larga escala) já foram discutidos ao logo deste trabalho e apresentados como características de um *middleware* para sistemas distribuídos convencionais.

Robustez é a capacidade de um determinado *middleware* se recuperar de falhas. Confiabilidade garante a equivalência entre a mensagem enviada e a mensagem que foi recebida. Tanto a robustez quanto a confiabilidade são propriedade comuns para um *middleware* transacionais.

Um *middleware* e dito "adaptativo"quando ele possui a capacidade de modificar sua estrutura para se enquadrar a um novo uso. Se determinado *middleware* for capaz de sentir o contexto e se auto-adaptar em tempo real às modificações do ambiente, ele é tido como *middleware* reflexivo.

O gerenciamento de contexto é fundamental para decidir as ações que serão tomadas com a detecção do contexto. Ele pode, por exemplo, expandir a capacidade de um dispositivo através da alocação de recursos no ambiente.

Projetar aplicações neutras aos dispositivos fornece interação transparente com o usuário. Tal transparência pode ser obtida através da definição de interfaces abstratas e predição de diferentes tipos de interação, levando ao envolvimento mínimo do usuário com problemas do sistema.

A invisibilidade é uma característica inerente da computação pervasiva, que está preocupada em manter o foco do usuário na aplicação propriamente dita e não nas ferramentas. O *software* deve aprender com o usuário, e permitir que ele mude suas preferências de uma maneira quase imperceptível.

Em nosso trabalho optamos por agregar propriedades da computação pervasiva ao *middleware* que estamos desenvolvendo. Além das propiedades básicas de descritas no início dessa seção (sensibilidade ao contexto, mobilidade e integração), em nosso trabalho temos o intuito de garantir as seguintes propriedades: interoperabilidade, escalabilidade e interação transparente com o usuário. Nos capítulos 4 e 5, descrevemos como cada uma dessas propriedades é garartida em nossa abordagem.

2.5 Interação Humano-Computador

O estudo das relações entre pessoas e computadores é denominado Interação Humano-Computador (IHC) [Dix et al. 2004]. Seu campo de estudo é multidisciplinar e envolve

a Ciência da Computação, Psicologia, Sociologia, Línguas, Artes, Design entre outras áreas de estudo. O foco de estudo da IHC não está nas pessoas e nem nos computadores, mas na interface entre os dois. Essa interface é comumente referenciada como interface do utilizador e é formada por artefatos de *hardware* e *software* que auxiliam as pessoas no processo de interação com o computador.

Neste trabalho, nossa preocupação é com uma subárea de estudo da IHC: acessibilidade. O conceito de acessibilidade, porém, é oriundo da arquitetura e posteriormente foi incorporado à área de IHC. Para melhor esclarecer o conceito de acessibilidade conforme pretendemos tratá-lo no contexto deste trabalho, o mesmo é formalizado a seguir.

2.5.1 Acessibilidade

Este termo é usualmente empregado para designar a capacidade de ser oferecido um determinado serviço a pessoas portadoras de necessidades especiais. Na área de arquitetura, o termo é usado para designar o quanto uma construção é acessível aos portadores de necessidades especiais [Piccolo et al. 2007], como paraplégicos, cegos, e outros. Para tornar uma construção acessível é preciso oferecer recursos, tais como rampas, barras de apoio, sinalizações sonoras, sinalizações visuais e outros recursos, que garantam o livre acesso desses usuários especiais ao ambiente.

Na grande área de Computação, o termo é estudado pela área de IHC (Interação Humano-Computador) e geralmente é utilizado para designar a capacidade de um determinado *software* em oferecer ferramentas que permitam que usuários com necessidades especiais possam interagir com a máquina [Dix et al. 2004]. O tipo mais comum de ferramentas dessa classe são as de síntese e reconhecimento de voz, que auxiliam os usuários cegos a interagir com o sistema.

Atualmente, dentro do conjunto de usuários com necessidades especiais, não estão incluídos apenas os portadores de debilidades físicas, mas também aqueles com debilidades de *software*, de *hardware* e de rede. Esta ampliação no escopo dos usuários com necessidades especiais vai ao encontro da definição de inclusão digital [Guerreiro 2006]. Em outras situações, o conceito de acessibilidade se mistura com conceito de projeto universal.

2.5.2 Projeto Universal

Projeto Universal, do inglês *Universal Design* [Erlandson 2007], ou ainda Projeto Total, é uma subárea de estudo da área de Design que enfoca o desenvolvimento de produtos, serviços e ambientes a fim de que sejam usáveis pelo maior número de pessoas possível, independente de idade, habilidade ou situação. Está diretamente relacionado ao conceito de sociedade inclusiva e sua importância tem sido reconhecida por governos, empresários e indústrias.

No Projeto Universal é estudada uma série de questões que geralmente não são abordadas em um projeto comum, pois é necessário considerar todas as possibilidades de uso por usuários muito diferentes. Isso inclui questões sociais, históricas, antropológicas, econômicas, políticas, tecnológicas, e principalmente de ergonomia e usabilidade.

2.5.3 Síntese e Reconhecimento de Fala

Síntese de fala [Schröder 2001] é o processo de produção artificial de fala humana. Um sistema desenvolvido para este propósito é denominado sintetizador de fala. Um sintetizador de fala pode ser implementado em *software* ou *hardware*. O principal método para síntese de fala é método Texto-para-Fala (*Text-to-Speach*) [Sproat 1997]. Um sistema de Texto-para-Fala converte texto em linguagem normal para voz e consiste de duas fases principais. A primeira é a análise do texto, onde o texto de entrada é transcrito em fonemas ou outra representação lingüística. Na segunda fase ocorre a geração de formas de onda da voz, onde a saída acústica é produzida destes fonemas. Estas duas fases são geralmente denominadas de síntese de alto e baixo nível, respectivamente.

Reconhecimento de fala [Jelinek 1998] é uma técnica computacional que permite construir dispositivos que transcrevam a fala em texto automaticamente. Em alguns aplicações essa transcrição é apenas uma etapa intermediária no processo de compreensão da fala, possivelmente terminando com ações em resposta ao que foi dito. Uma vez que a importância está na descoberta do sentido das palavras e não apenas na detecção de palavras é comum definir essa área de estudo como sendo Reconhecimento de Fala ao invés de Reconhecimento de Voz.

Quando definimos a *Interpercepção* enquadramos a mesma como um solução que promove acessibilidade. Isso foi definido pois permitimos que pessoas com diferentes recursos computacionais acessam um mesmo sistema. Além disso, é acoplado a estrutura da *Interpercepção* ferramentas de síntese e reconhecimento de fala, promovendo o acesso de pessoas com deficiência visual.

2.6 Considerações Finais

Neste capítulo foi apresentada uma série de conceitos que norteiam a compreensão deste trabalho. Esses conceitos demonstram que o nosso trabalho está inserido em três áreas da computação: Realidade Virtual, Sistemas Distribuídos e Computação Pervasiva. A Computação Pervasiva é conseguida graças ao estabelecimento de um sistema distribuído. A união da Computação Pervasiva com conceitos de Realidade Virtual, principalmente ambientes virtuais, é um tópico de pesquisa recente. Algumas pesquisas sobre a união destas duas áreas são apresentadas no capítulo 3.

Capítulo 3

Trabalhos relacionados

Alguns trabalhos na literatura já buscaram soluções para o mesmo problema abordado neste trabalho: a criação de aplicativos para ambientes computacionais heterogêneos. Esses ambientes computacionais heterogêneos são um cenário recorrente na área de sistemas distribuídos e área da computação pervasiva. Os principais conceitos dessas duas áreas da computação foram discutidos no capítulo anterior.

A Computação pervasiva é um tópico de pesquisa recente e vários sistemas de*middleware* e aplicações pervasivas tem sido propostos. Algumas dessas abordagens são apresentadas na primeira parte deste capítulo. Essas abordagens propõem implementações capazes de alcançar um conjunto de propriedades da computação pervasiva discutidas no capítulo 2.

Na segunda parte deste capítulo são apresentadas abordagens para solução de problemas (principalmente heterogeneidade) de computação distribuída ou pervasiva, direcionados a aplicações para jogos.

3.1 Aplicações de Middleware para computação Pervasiva

Os trabalhos apresentados nesta seção demonstram o estado da arte das pesquisas com computação pervasiva. Eles tratam principalmente da proposta de sistemas de *middleware* para solução de problemas oriundos da pervasividade. Esses sistemas de *middleware* propostos visam oferecer propriedades da computação pervasiva, mas focam principalmente na mobilidade e na sensibilidade ao contexto. Eles foram usados para nortear as nossas idéias a respeito de sistemas de *middleware* para computação pervasiva. É comum nesses trabalhos o uso do termo nó para representar dispositivos na rede distribuída. Outro termo comum é recurso, em analogia a um determinado serviço oferecido por algum nó da rede. Esse termos poderão ser empregados na descrição dos trabalhos agora discutidos.

3.1.1 Projeto Aura

O Projeto Aura [Garlan et al. 2002], tem como objetivo minimizar distrações do usuário, criando um ambiente que se adapta ao contexto e às necessidades do mesmo. Ele foi especificamente projetado para ambientes pervasivos envolvendo comunicações sem fio, computadores portáteis ou não, e para espaços inteligentes.

Para alcançar seu objetivo, o projeto Aura realiza busca de recursos disponíveis. O Aura também deveria ser capaz de fazer realocação inteligente de recursos e a captação de informação de forma inteligente.

Por exemplo, um funcionário está em casa fazendo os slides de uma apresentação. Essa apresentação será exibida para diretoria da empresa dentro daqui a meia-hora. Esse funcionário tem o Aura instalado em vários dispositivos que usa ao longo do dia, como seu *notebook*, o computador do trabalho, o seu celular e o computador da sala de reuniões. O celular do funcionário possui uma agenda. O celular dispara um lembrete de que a reunião onde ele fará a tal apresentação começa em 15 minutos. O funcionário salva o arquivo dos slides e inicia o desligamento do *notebook*. O Aura entra em funcionamento nesse momento: ele acessa a agenda do celular e verifica qual o próximo evento. Ao descobrir que é uma reunião, o Aura busca por arquivos no *notebook* relacionados a essa reunião. Ao encontrar esses arquivos, o Aura envia uma cópia do mesmo para o celular. Depois o Aura se encarrega de acessar a rede pelo celular e enviar os arquivos por *e-mail*. No computador do escritório, o Aura ao verificar a chegada do *e-mail*, baixa os arquivos em anexo. O Aura ainda verifica na agenda o próximo evento e o local do mesmo. Ao decobrir que o próximo é na sala de reuniões, o Aura se encarrega de enviar os arquivos para um computador no local dessa sala.

Infelizmente, ao tentar atacar vários aspectos da computação pervasiva, o projeto Aura não consegue fornecer sistemas de *middleware* com todas as qualidades de pervasividade. A tendência hoje é fazer sistemas de *middleware* e arcabouços para problemas específicos, mesmo usando um modelo geral como parâmetro. Assim como o Aura, pretendemos desenvolver uma abordagem capaz de interagir e integrar diversos dispositivos. Porém, ao contrario do Aura nós procuramos estabelecer uma abordagem com um conjunto mais restrito de propriedades da computação pervasiva.

3.1.2 AlfredO

AlfredO [Rellermeyer et al. 2008] é uma arquitetura de *middleware* que possibilita que o usuário interaja de maneira flexível com outros dispositivos eletrônicos. O *middleware* proposto fornece escalabilidade, flexibilidade, independência do dispositivo, segurança, eficiência e é de fácil administração.

O sistema incorpora três mecanismos principais: um modelo de distribuição de *software* baseado em serviço; uma arquitetura de serviço em multi-camadas e um modelo de apresentação independente do dispositivo.

Para mostrar a validade da arquitetura foram implementadas duas aplicações: MouseController e AlfredOShop. A primeira é um protótipo de uma aplicação para controlar espontaneamente telas de informações em um telefone móvel. Já a segunda é uma aplicação sensível ao contexto que realiza buscas por serviços de compras pela rede na qual está inserido. Num cenário de um shopping center, por exemplo, ele buscaria nas redes das lojas por serviços locais de compras. Ao encontrar tal serviço ele baixaria aplicativo no aparelho celular e faria a transação de compra sem ter de ir até loja.

O AlfredO apresenta uma implementação mais concreta e viável de um sistema pervasivo. Isso graças a restrição das propriedades garantidas pelo *middleware* proposto pelo

AlfredO. Contudo, a proposta do AlfredO é trabalhar apenas com dispositivos móveis. Já a nossa abordagem não busca restrigir o uso apenas de celulares para o desenvolvimento de aplicações, pelo contrário, queremos permitir o uso de vários dispositivos para acessar o nosso ambiente virtual.

3.1.3 Collaborative resource discovery

O trabalho de [Al-Jaroodi et al. 2008] aplica agentes colaborativos na descoberta de recursos físicos e computacionais. O arcabouço original proposto pelos autores define uma arquitetura hierárquica auto-organizavel. Este arcabouço dependia da exigência de pelo menos um nó estável (fixo). Um segundo arcabouço foi proposto.

No novo arcabouço todos os nós podem ser móveis. O arcabouço cria e mantêm dinamicamente os recursos. Para tal, cada grupo elege um líder. Ele é escolhido segundo algum atributo relevante ao sistema. No caso deste trabalho ou autores usam a menor taxa de mobilidade do grupo para estabelecer quem é o líder. Os nós de um grupo trocam mensagens entre si, mas todos passam pelo líder. O líder se comunica com os outros líderes.

Os autores infelizmente não apresentam nenhuma aplicação construída com o arcabouço. Mesmo assim o arcabouço proposto demonstra ser relevante para criação de ambientes pervasivos com grande heterogeneidade.

3.2 Aplicações de Jogos multiusuários

É notório perceber que na literatura recente de sistemas distribuídos, boa parte das abordagens propostas é aplicada na construção de jogos multiusuários. O mesmo vale para o cenário da computação pervasiva. Por isso nesta seção são discutidos uma série de trabalhos direcionados à criação de jogos capazes de serem executados em múltiplos dispositivos.

3.2.1 Cross-platform games

O *Cross-platform* [Han et al. 2005] é um motor para o desenvolvimento de jogos 3D acessíveis por diferentes plataformas. O desenvolvimento deste motor está ligado ao conceito de jogos multi-plataforma, que são jogos que possuem versões para diferentes plataformas de vídeo-game e até mesmo versão para PC. Deste modo, o objetivo do motor *Cross-plataform* é unir os jogadores de um jogo multiusuário e multi-plataforma em um ambiente de jogo compartilhado, mesmo que cada jogador esteja conectado por uma plataforma de jogo diferente. O jogador também tem flexibilidade para trocar de plataforma a qualquer momento, sem perder o jogo que ele havia desenvolvido. Se o jogador possui uma plataforma X e sempre acessa o jogo desta plataforma, mas ele viaja para um local onde só está disponível uma plataforma B, ele poderá continuar seu jogo normalmente da plataforma B. Quando um jogo é desenvolvido com este motor, ele adquire uma

infra-estrutura que permite que o mesmo seja executado em diferentes plataformas (modelos) de video-games, além de ser capaz de se interligar a um ambiente com múltiplos jogadores a partir de qualquer plataforma.

Além do motor *Cross-plataform* definir uma arquitetura de *software* para garantir a interligação dos jogos multi-plataforma, esta arquitetura usa um único servidor para tratar as informações enviadas por todos os usuários. Assim, ele garante o compartilhamento do ambiente de jogos entre todos os usuários. Esta arquitetura também define uma camada de *middleware* para comunicação entre o servidor e as diferentes plataformas de jogo.

O uso de um servidor unificado é semelhante à solução que é abordada no presente trabalho. Contudo, o *Cross-plataform* está focado no desenvolvimento apenas de jogos com interface 3D, ao contrário da nossa abordagem que trata da interligação de ambientes com versões em várias interfaces de visualização.

3.2.2 Jogos Universalmente Acessíveis

Com a grande diversidade de jogadores a usabilidade se torna uma questão importante. Os jogos universalmente acessíveis, é uma teoria para a concepção de jogos que são utilizados em qualquer plataforma e por qualquer usuário [Grammenos et al. 2009].

Em outras palavras, essa teoria corresponde ao conceito de design universal [Preiser & Ostroff 2001] aplicado no desenvolvimento de jogos. Um jogo feito de acordo com as regras do jogo universal tem apenas uma versão: o jogo tem uma versão (dita universal) que é adequada para qualquer usuário e para ser usado em qualquer plataforma. Isso estabelece regras restritivas que devem ser seguidas durante a concepção do jogo. Por exemplo, o jogo deve ser desenvolvido em uma linguagem de programação portável.

Esse modelo restritivo é a grande diferença do tipo de modelo que propomos em nosso trabalho. Ao contrario dos jogos universamente acessíveis, nossa abordagem buscaria fornecer diferentes versões de um mesmo jogo para fornecer acessibilidade. Cada versão seria adequada a uma plataforma.

3.2.3 Pervasive Games

A diferença de *hardware*, necessidades e preferências dos usuários em um ambiente computacional é o grande problema a ser resolvido em um sistema distribuído. Ao promover a integração de ambientes heterogêneos, podemos criar uma abordagem generalizada. Nos últimos anos, muitas abordagens têm surgido propondo a criação de sistemas pervasivos, a fim de reunir os usuários de dispositivos diferentes em um sistema distribuído. A proposta dos Jogos Pervasivos (*Pervasive Games*) é uma dessas abordagens.

Essa abordagem define um novo tipo de jogo através do alargamento do conceito de computação pervasiva para um ambiente de jogo [Benford et al. 2005]. Nos jogos pervasivos, dispositivos móveis e sensores permitem uma nova experiência de jogo, que estende o espaço (digital) de jogo para o mundo real. O foco principal do jogo é generalizar para permitir que uma nova experiência de jogo, criando um ambiente de realidade mista para os jogos.

3.2.4 Jogos *Cross-media*

Um gênero de jogo pervasivo chamado de jogos *cross-media* [Lindt et al. 2005] é centrada sobre a grande variedade de consoles de jogos, mobilidade e outros dispositivos que podem ser usados juntos permitindo uma nova experiência de jogo. O trabalho acima apresenta um estudo sobre os jogos onde os jogadores têm diferentes modos de jogo, que são fornecidos por diferentes dispositivos. Os jogadores podem se comunicar entre si através de SMS. O uso de um aparelho GPS permite aos usuários ver a posição do outro no ambiente de jogo. Os jogos *cross-media* são um gênero de jogo pervasivo, que visa a criação de uma abordagem de realidade mista para um ambiente pequeno de jogo.

3.2.5 Jogos PM2G

Ao integrar as idéias de jogos universais e jogos *cross-media*, os jogos PM2G (acrônimo de *Pervasive Mobile Multi-player Game*) [Trinta et al. 2006] definem cenários específicos para jogos universais que usam dispositivos diferentes. Para cada dispositivo utilizado, o PM2G propõe um ambiente de jogo diferente que é adequado para essa interface de dispositivo. Estes cenários também são adequados para a rede disponível, desde que cada um ofereça um cenário diferente de mini-jogo. O resultado de cada mini-jogo é arquivado causando a evolução do grande ambiente de jogo que é compartilhado e visto por todos os jogadores.

Ao contrário da nossa abordagem, os jogos PM2G empregam um ambiente de jogo diferente para cada dispositivo. Em cada amabiente de jogo estão disponíveis ações diferentes. Em nossa abordagem garantimos o mesmo ambiente e o mesmo conjunto de ações em cada dispositivo. A diferença entre os dispositivos está apenas na visualização do mesmo.

3.3 Middleware para ambientes virtuais

Nessa seção estão reunidas as abordagens mais condizentes com o nosso trabalho. As abordagens apresentadas agora são focadas na criação de um *middleware* para construção de ambientes virtuais multiusuários.

3.3.1 Continuum

O *middleware* Continuum [Tran et al. 2002] visa oferecer uma arquitetura aberta e extensível de serviços de *middleware* para ambiente virtuais *on-line* e de larga escala. Sua estrutura é orientada a objetos e criada totalmente na linguagem Java. Ele permite a criação de ambientes persistentes e o uso de objetos autônomos (objetos que não são controlados pelos usuários).

O Continuum oferece serviços de manipulação de objetos, manipulação de eventos, comunicação de grupos e suporte de rede. Sua arquitetura de comunicação é baseada no modelo cliente-servidor. Para descrição de objetos é usada uma meta-linguagem de descrição. Para manipulação desses em larga escala o serviço provido usa uma técnica

de replicação de objeto. Nessa técnica um objeto possui uma versão original chamada de mestre e várias cópias escravas. As cópias são capazes de se comunicar com o mestre e avisar sobre uma alteração e garantir a alteração das demais.

Apesar de apresentar um conjunto de serviços bem definidos e passiveis de extensão, o Continuum só permite a criação de aplicações em Java. Aplicações criadas em outras linguagens não podem usufruir dos seus serviços, ao contrario da nossa abordagem que permite a criação de aplicações em outras linguagens.

3.3.2 CTU

O *middleware* CTU [Yatsu & Shibata 2009] permite a criação de ambientes virtuais colaborativos e imersivos. Seu objetivo é oferecer um conjunto de serviços que permita a criação de ambientes mais imersivo e amigáveis.

Os serviços do CTU permitem a captura e transmissão de áudio e vídeo em tempo real. Outro diferencial é o uso de sensores de realidade virtual. Os sensores promovem a imersão proposta. Os sensores pervistos na arquitetura básica do sistema são controles do *Nintendo Wii*, câmeras de vídeo e microfones.

O CTU oferece suporte ao uso de PC apenas. Além disso, ele foi desenvolvido para aplicações multiusuário de pequena escala. Os autores também não elucidam o que significa a sigla CTU.

3.4 Comparação entre os trabalhos

A Tabela 3.1 mostra um quadro comparativo entre os trabalhos discutidos na seção de Aplicações para jogos multiusuários. Os trabalhos foram agrupados nas linhas, sendo a última linha destinada à nossa abordagem, o GATE. Deste modo, buscamos traçar este comparativo entre as várias abordagens voltadas a jogos.

A Tabela 3.1 compara as abordagens através de seis questões principais: Dispositivos, isso significa que o acesso através de diferentes dispositivos é garantido pela abordagem; interface visual é relacionada com o tipo interface (visual) de saída; Rede, define os diferentes tipos de redes permitidas pela abordagem; interoperação, que é a possibilidade de uma abordagem permitir que um aplicativo cliente seja construído em qualquer plataforma, o que permite que o ambiente seja executado em qualquer dispositivo; persistência, que significa a possibilidade de o ambiente armazenar dados sobre os acontecimentos do passado e criar um ambiente compartilhado persistente para os usuários; acesso (ou acessibilidade) que informa se a abordagem fornece ferramentas para permitir que os usuários com necessidades especiais possam utilizar o ambiente.

Como podemos ver na Tabela 3.1, a nossa abordagem, prevê o uso de qualquer dispositivo para acesso a uma aplicação multiusuário, ao contrário da maioria, que foca seu uso em aplicações para celulares e PC. As duas primeiras abordagens da tabela ainda procuram estabelecer o uso de aparelhos de Realidade Aumentada (*AR System*). Nessas abordagens o ambiente pervasivo é montado para criação de jogos de Realidade Mista. Mesmo que esse não seja o foco da nossa abordagem, nada impede que Arquitetura de *software* apresentada neste trabalho seja usada para tal aplicação. No quesito interface

Abordagem	**Dispositivos**	**Interface**	**Rede**	**Persistência**	**Inter-operação**	**Acesso**
Per. Games	PC, celular, AR system	3D, 2D, textual	LAN, WAN	não	não	não
Cross-media	PC, celular, AR system	3D, 2D, textual	LAN, WAN	não	não	não
PM2G	PC, celular	2D e textual	Internet, LAN, WAN	sim	não	não
Jogos Universais	não se aplica	não	Internet	não	não	sim
Cross Plataform	Consoles de video-game	3D	Internet, LAN, WAN	sim	não	não
GATE	PC, celular, AR system	3D, 2D, textual	Internet, LAN, WAN	sim	sim	sim

Tabela 3.1: Comparação entre os trabalhos relacionados sobre jogos

visual podemos perceber que outras abordagens (*Cross-media* e *Pervasive Games*) também trabalham com diferentes reapresentações da informação, assim como a nossa. Essa qualidade é importante pois determinados dispositivos podem não ser capazes de executar uma interface 3D. Na propriedade de Rede está sinalizado que interação em um ambiente pervasivo passa também por uma diversidade de tipos rede. A nossa abordagem e as abordagens *Cross-Plataform* e PM2G oferecem uma infra-estrutura capaz de fornecer acesso através de um conjunto maior de redes. Nossa abordagem ainda permite a criação de aplicações para redes *Bluetooth*. A propriedade de persistência é importante para criação de jogos multiusuários *on-line* na atualidade. Na verdade essa qualidade já era importante na época de criação do MUD. Como o MUD é baseado em jogos de RPG, é intrínseca à natureza desse jogo a evolução dos personagens. Isso pode ser conseguido com o armazenamento de informações sobre as ações do jogador. Para tal basta adicionar a estrutura do sistema um banco de dados. Essa qualidade é atendida pela nossa abordagem, e pelas abordagens *Cross-Plataform* e PM2G. Isso demonstra ser o nosso interesse em garantir que nossa abordagem permitirá a criação de jogos, apesar de não diretamente o seu foco. O GATE, a nossa proposta, é a única abordagem que atende Interoperabilidade. Por isso, nossa abordagem permite a criação de aplicativos capazes de se comunicar independente do dispositivo em que são executados. Essa comunicação ainda é feita de uma forma que a interação seja transparente. A transparência durante a interação é também um dos mecanismos que conferem acessibilidade a nossa abordagem. A proposta dos jogos universalmente acessíveis permite o uso de qualquer dispositivo e também fornece ferramentas de acessibilidade. Contudo, essa abordagem consegue isso através de uma única versão do aplicativo final, e não através de inter-operação de vários aplicativos.

Na Tabela 3.2 traçamos um outro comparativo, agora entre os trabalhos de *middleware*

para ambientes virtuais. Eles são comparados pelas mesmas propriedades usadas na comparação com os jogos multiusuários.

Como ilustra a Tabala 3.2, tanto o *Continuum*, como o CTU trabalham apenas com aplicações para PC, ao contrário da nossa, que visa outros dispositivos, além do PC. Na propriedade de Rede a nossa abordagem e a *Continuum* oferecem uma infra-estrutura capaz de fornecer acesso através de um conjunto maior de redes. Nossa abordagem ainda permite a criação de aplicações para redes *Bluetooth*, que não permitido pelo *Continuum*. A propriedade de persistência é atendida pela nossa abordagem, e pela abordagem *Continuum*. Tanto *Continuum*, quanto o CTU trabalham com interface 3D apenas. A nossa abordagem é novamente a única que se preocupa com as prerrogativas de acessibilidade.

Abordagem	**Dispositivos**	**Interface**	**Rede**	**Persistência**	**Inter-operação**	**Acesso**
Continuum	PC	3D	Internet, LAN, WAN	sim	não	não
CTU	PC	3D	LAN	não	não	não
GATE	PC, celular, AR system	3D, 2D, textual	Internet, LAN, WAN	sim	sim	sim

Tabela 3.2: Comparação entre os trabalhos relacionados sobre *middleware*

3.5 Considerações Finais

Os trabalhos apresentados neste capítulo mostram que a principal dificuldade na construção de uma aplicação pervasiva está na adoção de propriedades ligadas a esse tipo de aplicação. Por isso, alguns trabalhos buscam restringir a quantidade de propriedades garantidas pela aplicação. A propriedade colocada como sendo a principal, por quase todos os trabalhos, é a mobilidade. Esses trabalhos buscam atendê-la através da implementação de sistemas de*middleware* para permitir a interação de dispositivos móveis com outros dispositivos. Neste capítulo, também ficou evidente que a diversidade de um ambiente pervasivo não está apenas na diversidade de dispositivos. Existe também a diversidade de redes e a diversidade perfis de usuários. A resolução desses problemas de diversidade está atrelada a definição de uma arquitetura de *software* capaz de gerenciar a heterogeneidade do sistema. Por isso, no capítulo 4 apresentamos uma formalização destes problemas. Nesse mesmo capítulo apresentamos algumas arquiteturas voltadas ao tratamento dessas questões.

Capítulo 4

A Interpercepção

Para que dispositivos de diferentes plataformas de *hardware* e *software* possam se comunicar (ou seja, trocar dados e entender bem a semântica desses dados) é necessário desenvolver um modelo que reduza a complexidade inerente à computação distribuída. Em resposta a esse desafio de comunicação, que é uma conseqüência do problema da heterogeneidade dos ambientes de computação distribuída, algumas possíveis soluções são propostas na literatura: RPC, CORBA, MOM, entre outras. Essas soluções foram discutidas no capítulo 2. Neste capítulo introduzimos a nossa abordagem, a *Interpercepção*.

Antes de discutirmos a nossa abordagem é preciso categorizar o nosso problema. Esse problema era inicialmente a diversidade de interfaces de visualização. Contudo, decidimos ampliar a *Interpercepção* para permitir a criação de ambientes virtuais para vários dispositivos. Surge um problema maior: a heterogeneidade de um sistema distribuído.

4.1 Heterogeneidade de um sistema distribuído

Como já discutido anteriormente, a computação distribuída tem como seu principal problema a heterogeneidade do ambiente no qual estão inseridas as suas aplicações. No inicio das pesquisas dessa área a heterogeneidade se concentrava na diversidade de plataformas. Dentro do escopo de plataformas está a diversidade tanto de *hardware* quanto de *software*. O desenvolvimento das pesquisas na área de redes de computadores fez surgir novos modelos de conexão, conferindo um aspecto de diversidade no âmbito da infra-estrutura de conexão. As pesquisas na área de IHC forneceram evidências sobre a diversidade de perfis de usuários. Em resumo, um exemplo a heterogeneidade de um sistema distribuído é ilustrado na Figura 4.1.

Como mostrado na Figura 4.1, o ambiente aqui proposto prevê a integração de vários dispositivos comuns ao dia-dia (telefones celulares, PCs e Set-top Box). O sistema apresentado nessa figura é baseado na conexão de todos os dispositivos com a *Internet*. Contudo, a forma como esses dispositivos acessam a *Internet* é diferente. Essa diferença está evidenciada pelo uso de setas distintas para conexão entre o dispositivo e a *Internet*. Além disso, está prevista a interação de pessoas de diferentes características, como diferentes faixas etárias (adolescente, homem e mulher adultos) e diferentes necessidades (pessoa cega). Com isto, o sistema deverá fornecer diferentes formas de interação (setas que ligam cada usuário ao dispositivo que ela está usando) e visualização para os usuá-

rios. Observe que o ambiente ilustrado na Figura 4.1 nos faz lidar com as três camadas de heterogeneidade existentes nesse domínio de aplicação:

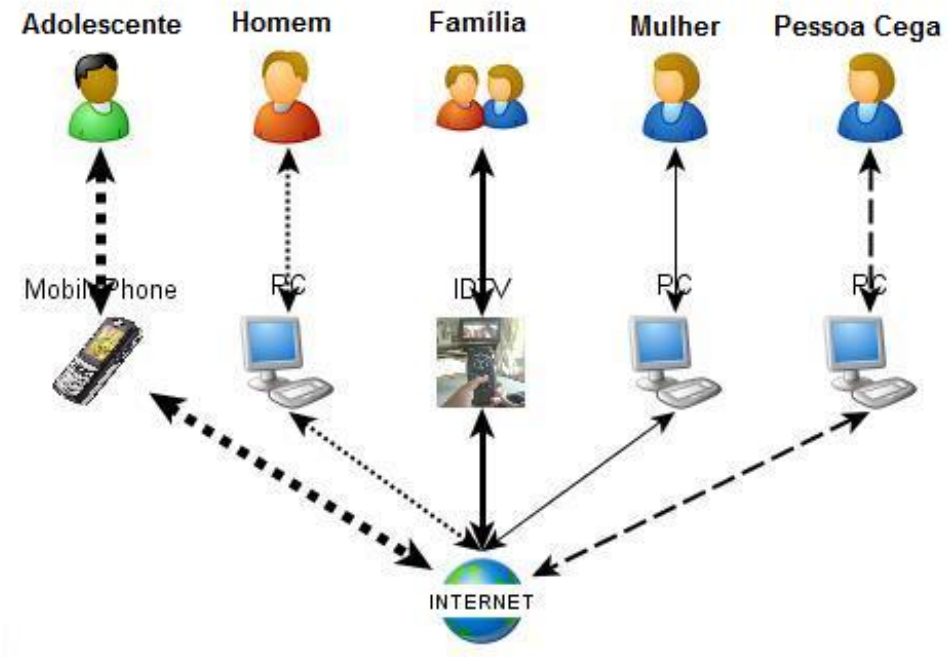

Figura 4.1: Ambiente heterogêneo para integração entre dispositivos

- **Perfis de usuário**: diz respeito aos diferentes tipos de usuários que acessam o sistema. Eles são diferentes no tocante a idade, cultura, idioma, necessidades especiais, etc. Cada usuário é único e tem necessidades diferentes. Como exemplo, alguns conteúdos são proibidos para um adolescente ou uma criança. Uma interface com a síntese e reconhecimento de fala é exigida para uma pessoa cega. Uma família ou um grupo de usuários compartilhando um dispositivo ao mesmo tempo exige diferentes formas de interação. Um homem dos Estados Unidos precisa de línguas e ícones diferentes que uma da mulher do Japão. Em outras palavras, a acessibilidade, a idade e os aspectos culturais devem ser considerados, se um sistema é destinado a desenvolver uma boa interface. Assim, cada usuário precisa de uma interface personalizada de acordo com suas necessidades e/ou interesses (os interesses de um adolescente são diferentes do interesse de uma família).
- **Camada de Dispositivos**: os dispositivos utilizados para acessar o sistema são distintos quanto ao poder de processamento (um computador pessoal tem uma capacidade de processamento maior que um celular), modo de interação (*joystick*, controle remoto, *mouse*, teclado, luva de dados, etc.), modo de exibição (exibição por texto ou com gráficos 2D, 3D), etc. Dependendo dos recursos financeiros, a ocasião e as preferências do usuário a escolha de um dispositivo em vez de outro podem ser diferentes para cada usuário. Isso gera um cenário heterogêneo onde *smart phones*, telefones celulares, computadores pessoais, *set-top boxes*, PDAs e

outros dispositivos com recursos de *hardware* (processador, dispositivos de entrada / saída, memória) têm sido utilizados.

- **Camada de Redes**: a diversidade de um sistema distribuído pode ainda diferir quanto à rede de transmissão de dados que os usuários estão utilizando para acessar o sistema. Essa camada considera protocolos e recursos de *hardware* usados por cada dispositivo. Exemplos da diversidade de requisitos a partir desta camada, é demonstrado pelo fato que os usuários poderiam estar conectados por banda larga, GPRS, conexões *dial-up*, ou *bluetooth*.

Resolver os problemas de cada camada de uma única vez é uma tarefa árdua e complexa. Para reduzir essa complexidade é viável concentrar em resolver o problema de uma camada e presumir que as demais se encontram num estado ideal. Por exemplo, resolver a questão da diversidade de dispositivos, presumindo que sistema está conectado por um mesmo tipo de rede e a aplicação já está adequada a todos os usuários (ou pelo menos é adequada ao seu publico alvo).

No aspecto da diversidade de dispositivos, modelos baseados em RPC ou MOM foram propostos e alcançaram ressultados satisfatórios. Para diversidade de perfis de usuários desenvolvemos anteriormente a *Interpercepção*. Essa abordagem foi apresentada no capítulo 1 de forma sucinta. Ela será discutida agora com mais detalhes.

4.2 Arquitetura da *Interpercepção*

O funcionamento da *Interpercepção* como descrito até o momento não estabelece nenhuma arquitetura para sistemas distribuídos. Contudo, para validação da mesma foi proposta uma arquitetura baseada no modelo cliente-servidor [Dantas et al. 2006]. Essa arquitetura, chamada de arquitetura *Interdimensional* (e abreviada para InterD) é apresentada na Figura 4.2.

Na Figura 4.2 os vários aplicativos que eram previstos no cenário da *Interpercepção* são agora representados por clientes. Cada cliente é um componente de *software* com suporte a um tipo de interface de visualização: textual (aqui representado como 1D), 2D ou 3D. Todos os clientes se conectam ao mesmo servidor. Esse servidor tem acesso ao banco de dados onde está armazenado o ambiente. Mesmo que a Figura 4.2 não apresente uma representação formal do banco ele continua subentendido como sendo parte do sistema.

A distinção entre as interfaces das aplicações clientes está na dimensão dos gráficos suportados, ou seja, uma aplicação cliente possui interface 2D, outro possui interface 3D, assim por diante. Por isso na especificação da *Interpercepção* esta arquitetura foi chamada arquitetura *Interdimensional* (ou InterD).

Os clientes representados na arquitetura *InterD* são a junção de todos os clientes com um determinado tipo de interface de visualização. Deste modo, o Cliente 2D nesta figura representa todos os usuários que utilizam uma aplicação cliente com interface de visualização 2D. Essa arquitetura também demonstra que aplicações clientes diferentes acessam o mesmo servidor e, portanto, trocam as mesmas informações. Se por exemplo, uma mensagem de texto é enviada por algum cliente 3D, ela poderá ser repassada para os

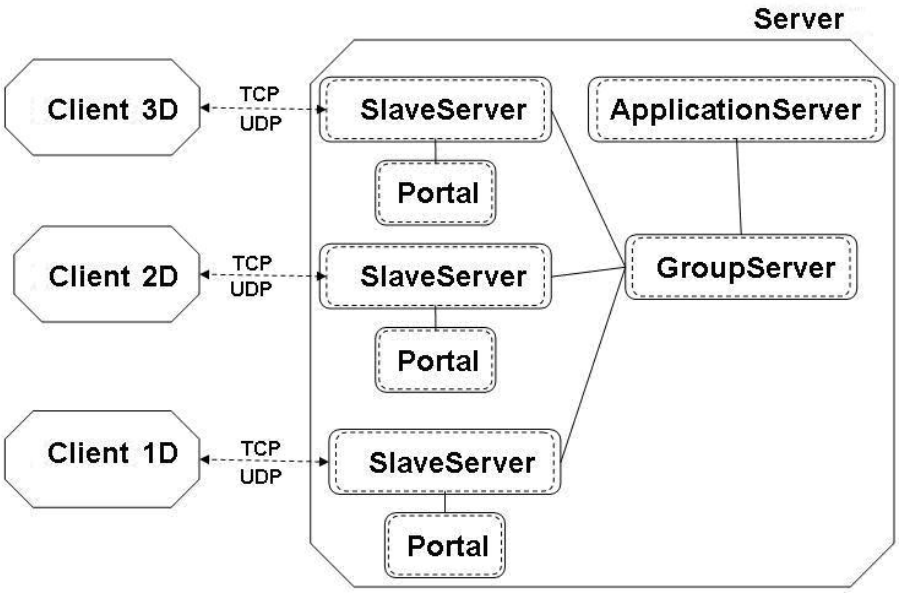

Figura 4.2: Arquitetura Interdimensional (InterD).

demais tipos de clientes. Essa arquitetura ainda estabelece que a comunicação entre os clientes e o servidor pode ocorrer através de protocolo UDP e TCP.

O servidor apresentado na arquitetura *InterD* é formado por quatro componentes de *software*: *SlaveServer*, *GruoupServer*, *ApllicationServer* e *Portal*. Os três primeiros componentes são oriundos do arcabouço H-N2N [Burlamaqui et al. 2006], que serão detalhados na próxima seção.

4.3 O Arcabouço H-N2N

O H-N2N (abreviatura de *Hierarquical* N *to* N) é o arcabouço de *software* para aplicações distribuídas e massivas. Esse arcabouço é baseado no modelo cliente-servidor e tem como principal característica o estabelecimento de uma rede hierárquica de servidores. Essa rede hierárquica permite aumentar gradativamente a quantidade de clientes que são atendidos. A Figura 4.3 ilustra o funcionamento de H-N2N.

Como ilustra a Figura 4.3, no nível 0 está a raiz da hierarquia de servidores, representada por um *ApllicationServer*. Esse componente cria um conjunto de grupos (representado pelo *GroupServer*) para reunir os clientes (representado na Figura 4.3 como *UserClient*). Se por exemplo, um ambiente virtual é formado por um conjunto de salas, cada sala teria um *GroupServer*. Para cada cliente que se conecta ao ambiente é criado um *SlaveServer*. Esse *SlaveServer* está ligado a um *GroupServer*. No momento da conexão, esse *SlaveServer* está ligado ao grupo que atende as requisições da sala de entrada do ambiente.

À medida que novos clientes se conectam chega um momento que o servidor principal (**ApllicationServer**) não é mais capaz de atender as requisições. Esse servidor principal então elege uma máquina cliente para ser um novo servidor. Ele envia uma instância do

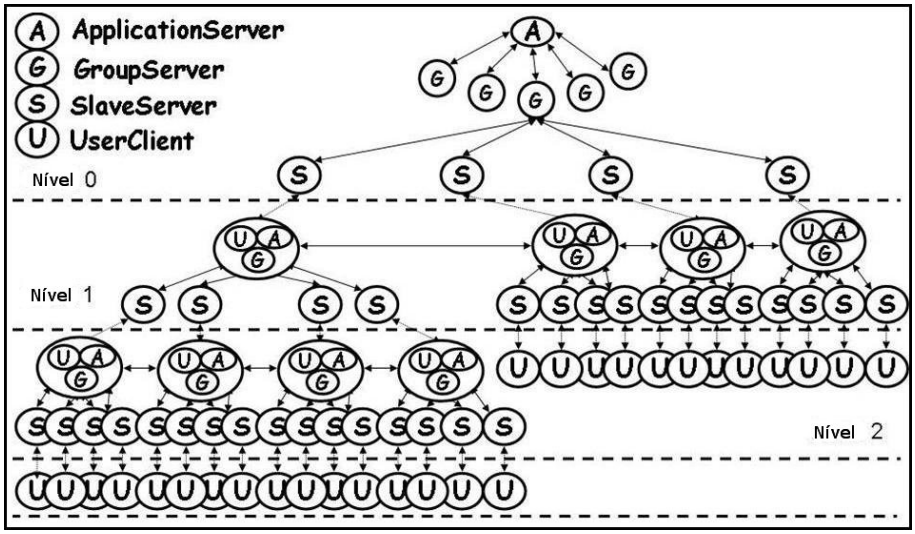

Figura 4.3: Exemplo de configuração do H-N2N.

ApllicationServer para essa máquina. Essa instância dá início a um novo processo de criação hierárquico de oferta de serviço. Isso cria o nível 1 da Figura 4.3. Assim, uma máquina que antes possuía apenas uma cópia de *UserClient*, agora possui também uma cópia de *ApllicationServer* e *GroupServer* também. Essa máquina agora é um servidor. Ao receber conexões ele cria instâncias de *SlaveServer* para atender aos clientes. Esse processo continua indefinidamente para aumentar a escala de clientes conectados.

Na especificação do arcabouço H-N2N, a complexidade do tratamento de requisições é diluída através da criação de camadas de serviço. Essas camadas são representadas pelos componentes de *software* que formam o H-N2N.

O *SlaveServer* mantém uma comunicação com o cliente. Existe um componente deste tipo para cada cliente. Ele representa a camada mais baixa na hierarquia de servidores. Ele é capaz de tratar as requisições de um cliente, desde que elas não envolvam a comunicação com outros clientes. Caso o *SlaveServer* receba alguma requisição que gere comunicação com outros clientes, ele passa a requisição para o *GroupServer*.

O *GroupServer* é um componente que contém a lista dos *SlaveServer*'s que representam os clientes conectados. Ele é o responsável por gerenciar as trocas de mensagens entre os clientes de um determinado grupo. Esse componente também responde pelo processamento das operações de filtragem e junção. Ele também pode se comunicar diretamente com outros *GroupServers* de mesmo nível.

Na camada mais alta da hierarquia está o *ApplicationServer*. Ele é um componente responsável pela espera das conexões dos clientes. Ele atua como um servidor principal. Esse componente é capaz de enviar uma mensagem a todos os clientes conectados. Quando uma mensagem deve ser enviada todos os clientes, ela feita pelo *ApplicationServer*.

Além dos componentes do H-N2N, existe no lado servidor da arquitetura InterD um quarto componente chamado **Portal**. Este componente não pertence originalmente ao H-N2N e trata-se de uma modificação necessária para a concepção da *Interpercepção*.

4.4 O componente Portal

O Portal é responsável pelo processo de conversão das mensagens que são trocadas entre os clientes e o servidor. Para realizar as comunicações com as várias aplicações clientes previstas, é necessário definir protocolos de comunicação entre o servidor e os vários tipos de clientes. Além disso, como cada cliente possui uma interface distinta para visualização, é necessário converter a informação que vai de um cliente para o outro.

Um cliente com interface 2D não é capaz de visualizar dados 3D. Contudo, para que eles possam interagir e compartilhar informações é necessário que as informações vindas de um cliente 3D sejam exibidas para o 2D. Para que tal situação ocorra é necessário transformar os dados que vão do cliente de um tipo para um cliente de outro tipo. Portanto o processo é necessário para conversão de e para qualquer interface de visualização.

Para que ocorra a *Interpercepção* é preciso que exista um mecanismo de conversão de dados de um domínio de aplicação para outro. Esse mecanismo foi construído na forma de um componente de *software* que foi acoplado a arquitetura InterD. Este mecanismo foi chamado de Portal, por ser um meio de conexão entre mundos distintos, representada pela diferença das interfaces de visualização com gráficos de diferentes dimensões geométricas (1D ou textual, 2D e 3D).

Para ilustrar o funcionamento do Portal, suponha que um usuário conectado através da interface 3D, desloca-se no ambiente. Esse usuário geraria um fluxo de mensagens de movimento contendo informações como identificador do usuário e sua posição nas coordenadas x, y e z. Assim que tais mensagens chegam ao servidor elas serão repassadas para todos os *SlaveServer*. O *SlaveServer* então se encarrega de enviar as mesmas para os seus clientes. Porém, com a adição do componente **Portal** à arquitetura *InterD*, algumas mensagens passaram por uma função de conversão. Com isso o dado transmitido passa a se adequar ao seu cliente de destino.

A função de conversão está implementada no **Portal**. Quando esse componente recebe uma mensagem, ele verifica quais são as interfaces de origem e de destino, e, caso sejam diferentes, ele realiza a conversão adequada antes de repassar a mensagem. A informação que chega ao cliente de destino é condizente com a interface de visualização suportada por ele. Essa informação pode ser oriunda de qualquer outro tipo de cliente. Como ela foi convertida não existem meios de identificar o cliente de origem. Por isso dizemos que a comunicação entre os clientes na *Interpercepção* ocorre de forma transparente.

Agora que o Portal foi explicado, o entendimento sobre a *Interpercepção* está completo. É possível sintetizar agora a *Interpecepção* em princípios fundamentais, definidos como as leis da *Interpercepção*.

4.4.1 As Leis da *Interpercepção*

Para desenvolver um sistema com suporte a *Interpercepção*, é necessário seguir as seguintes regras:

- **Correspondência de Ambientes**: Cada ambiente possui um modelo tridimensional, uma versão correspondente em planta baixa (ou uma projeção isométrica), e

uma versão em descrição textual. Isso permite que em cada versão o mesmo ambiente seja visualizado, porém de uma forma adequada à interface de visualização de cada aplicação;

- **Abstração de Mensagens**: Os usuários ao se conectarem a qualquer uma das interfaces conseguirão perceber mutuamente os demais de uma forma síncrona. Isso ocorre graças ao gerenciamento da navegação ser centralizado em um único servidor. Este servidor trata em um nível de abstração mais alto os dados referentes à movimentação e ao fluxo de usuários;
- **Adequação de Interface**: Nenhuma interface de visualização é capaz de suportar dados de outra interface. Por isso, para o usuário final, as informações exibidas devem ser adequadas ao tipo de interface de visualização a qual ele está acessando.

A lei da correspondência é garantida pelo principio original de funcionamento da *Interpercepção*, onde temos o ambiente representado por um arquivo de meta-dados. Com base neste arquivo pode ser construídas as várias versões do ambiente. Mesmo que diferentes na representação gráfica, cada versão exibe o mesmo conjunto de informações. Então se no arquivo de meta-dados existe uma cadeira azul, essa cadeira será representada em 3D, 2D e descrita em forma de texto.

A lei da abstração é garantida pela utilização de servidor central que trata as mensagens dos vários tipos de cliente. Essa lei é estabelecida pelo uso da arquitetura *InterD*. Mesmo que na arquitetura esteja representado apenas um servidor, é possível utilizar vários servidores interligados. Isso é possível graças ao uso do arcabouço H-N2N que permite a criação de sistemas para uma larga escala de clientes.

A lei da adequação é garantida pelo **Portal**. Ele possui funções que transformam a informação de uma dimensão geométrica para outra. Assim, cada cliente já recebe a informação adequada ao seu tipo de interface. A lei adequação também confere acessibilidade ao sistema. Uma vez que a mesma oferece um meio de reunir usuários com diferentes requisitos de software, ela indiretamente confere acesso a diferentes perfis de usuário.

A lei da adequação abre margem para outras formas de acessibilidade. Ela permite que a informação possa ser transformada para se adequar à necessidade de um usuário com deficiência visual, por exemplo. Essa adequação é possível através da criação de um cliente capaz de realizar Reconhecimento e Síntese [Schröder 2001] de Fala. Na verdade usando a *Interpercepção* bastaria adicionar ao cliente textual, um componente responsável por realizar o Reconhecimento e Síntese de Fala.

Para interligar ambientes virtuais que possuem interfaces de visualização com dimensões gráficas distintas, nós propomos uma teoria que denominamos anteriormente de interdimensionalidade. Essa teoria trata da interligação desses ambientes com dimensões diferentes (por isso o nome interdimensionalidade). A proposta dessa teoria é converter as mensagens que são trocadas entre os clientes do sistema, quando os clientes possuem interfaces com dimensões distintas. Como resultado dessa conversão, os usuários são capazes de perceber uns aos outros de forma indistinta.

O processo de conversão entre as dimensões é o que garante também a lei adequação da *Interpercepção*. Contudo, a lei adequação abre margem para outras formas de adequação, além da adequação de dimensões. Essa lei poderia permitir a tradução de texto entre diferentes idiomas ou mesmo a tradução de gestos entre diferentes culturas. Todas essas

traduções ou conversões têm o intuito de adequar a informação ao usuário (e também a aplicação cliente) de destino. Por isso, dentro da lei de adequação, optamos por definir graus de adequação.

4.4.2 Graus de Adequação da *Interpercepção*

O conceito de adequar informações é abrangente. Com o intuito de formalizar esse processo de adequação das interfaces de visualização de uma aplicação foi criado uma lista de graus de *Interpercepção*. Eles foram definidos da seguinte forma:

- **1º Grau de Adequação**: Tradução de informações entre representações gráficas de dimensões distintas. A representação textual é tida por definição como sendo de dimensão unidimensional (1D). Essa definição advém de uma analogia com o sinal sonoro, que é unidimensional. Nesse sinal informação transportada pode ser qualquer informação que poderia normalmente ser representada por um texto. Por isso decidimos definir o texto como sendo um sinal 1D.
- **2º Grau de Adequação**: Uso de ferramentas de acessibilidade. Por exemplo, ferramentas de Reconhecimento de Síntese de Fala para atender as necessidades de usuários portadores de necessidades especiais.
- **3º Grau de Adequação**: Tradução de texto entre idiomas, durante a interação dos usuários no ambiente. Isso permite a comunicação dos usuários sem barreiras lingüísticas.
- **4º Grau de Adequação**: Tradução de gestos entre culturas diferentes. Essa tradução é pensada para comunicação através de videoconferência ou mesmo para comunicação gestual através de avatares. Seu intuito é quebrar barreiras culturais dentre de um ambiente virtual.

Os graus de adequação foram definidos com base criação das metas da *Interpercepção*. Nossa meta original era traduzir informações entre representações gráficas de dimensões distintas. Por isso esse é o primeiro grau de adequação. Depois incluimos as fermentas de acessibilidade, criando o segundo grau, e assim por diante.

4.5 A *Interpercepção* no escopo de múltiplos dispositivos

No início do capítulo separamos os problemas da computação distribuída em três camadas. As propriedades de *middleware* e modelos de computação distribuída apresentadas no capítulo 2 são capazes de resolver os problemas apontados nas camadas de dispositivos e redes. As propriedade dos sistemas de *middleware* pervasivos, interação transparente e invisibilidade, são capazes de resolver a heterogeneidade da camada de perfis de usuários.

Neste trabalho, propomos uma nova arquitetura de *software* para a *Interpercepção* que permita um determinado aplicativo multiusuário *on-line* para ambientes virtuais, ser executado por diferentes clientes em dispositivos distintos. Esta abordagem cria um sistema distribuído pervasivo onde a aplicação proposta estará disponível para os usuários a

todo tempo e em todo lugar. Esse requisito também atrela a diversidade de rede ao ambiente. A Figura 4.4 ilustra a arquitetura proposta. Demos a esta arquitetura o nome de Interpecepção entre dispositivos (abreviada para InterPD).

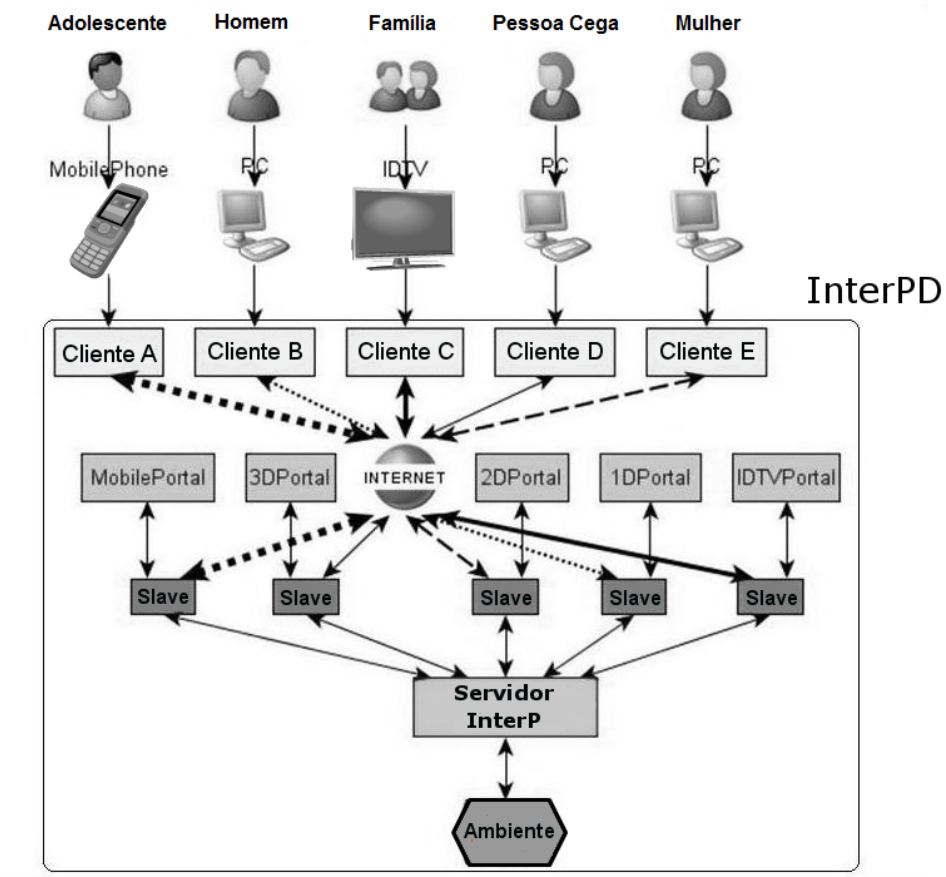

Figura 4.4: Arquitetura para *Interpercepção* entre dispositivos

Na Figura 4.4 é possível ver que a estrutura da *Interpercepção* entre dispositivos é organizada em cinco camadas. Na camada mais baixa temos o ambiente, que pode ser um ambiente virtual ou ambiente de jogo. Independente de que ambiente seja, esta camada representa o serviço principal fornecido pelo servidor. Logo acima do ambiente está o Servidor **InterP** (abreviatura de *Interpercepção*). Esse servidor é construído com base no arcabouço H-N2N. Nesse servidor estão embutidas estruturas comuns ao arcabouço, como o *ApplicationServer* (servidor principal com acesso direto a aplicação) e o *GroupServer* (servidor de grupos).

Na arquitetura da Figura 4.4, acima do servidor está a camada de *SlaveServers*, abreviado apenas para *Slave*. Esse conjunto de módulos é derivado da estrutura do **H-N2N**. Como o arcabouço **H-N2N** é usado na construção do servidor, ele prevê o uso dos módulos *Slave* para tratar das conexões diretas com os clientes.

Os usuários acessam o ambiente através de uma aplicação cliente, representado na arquitetura da Figura 4.4 pelos módulos com o nome Cliente (de *A* até *E*). Está representado

um cliente diferente para cada usuário para representar a diferença de uso ou interesse de cada usuário.

A separação entre o lado cliente e servidor na Figura 4.4 está na camada onde está posicionada a *Internet*. A *Internet* atua nesse cenário como intermediário entre os dois lados. Através da *Internet* surge à infra-estrutura de rede com diferentes modos de conexão (simbolizado por diferentes tipos de setas na figura 4.4). Essas conexões fluem através da rede interligando os aplicativos clientes com os *Slaves*.

Na mesma camada da *Internet* estão localizados os Portais. Esses módulos previstos pela definição da *Interpercepção* atuam na conversão de informações entre clientes diferentes. Os Portais garantem a lei de adequação da Interperecepção, descrita anteriormente no capítulo 1. Eles estão posicionados no lado do servidor, ligados ao *Slave*. Essa estrutura é similar a arquitetura **InterD** apresentada anteriormente no capítulo 1. Porém, na nova arquitetura da figura 4.4 está previsto a presença de dispositivos diferentes e também conexões distintas.

O Portal surge agora como um componente especializado, diferindo no tipo de cliente para o qual ele se dispõe a converter as informações. O serviço antes oferecido pelo Portal era adequar dados que seriam exibidos na interface de um cliente associado ao Portal. O Portal continua oferecendo esse serviço, porém a um conjunto maior de clientes. Na verdade, o intuito agora é garantir uma infra-estrutura que permita atender a vários (outros) tipos de clientes.

4.6 Considerações finais

A computação distribuída e também a pervasiva possuem ainda muitos problemas não resolvidos, principalmente devido a uma série de desafios trazidos por fatores tais como a variedade de equipamentos, dispositivos, sistemas de *hardware* heterogêneos, sistemas operacionais distintos, possíveis diferenças na arquitetura de redes, entre outros. Talvez o maior destes desafios seja tornar comunicáveis dispositivos distintos, provavelmente com sistemas operacionais e arquiteturas distintas e com aplicações desenvolvidas em linguagens diferentes. Transpor essas barreiras permitiria que esses dispositivos trocassem dados entre si e fossem, portanto, capazes de se comunicar.

Para a realização do presente trabalho foi desenvolvido uma abordagem baseada em *middleware* que se insere na categoria dos Middleware Orientados a Mensagens. A especificação da nossa abordagem e os serviços associados ao mesmo, chamada de GATE será discutida no próximo capítulo.

Capítulo 5

GATE

A meta principal da nossa abordagem é fornecer serviços para aplicativos voltados à criação de ambientes virtuais. Por isso dizemos que o novo intuito do componente Portal é "alcançar todos os ambientes". Em inglês, essa frase foi traduzida como "get all the environments" produzindo a sigla *GATE* que denomina a nossa abordagem, baseada em *middleware* e formado pela reunião dos Portais previstos nas arquiteturas do capítulo anterior. Além disso, o *middleware* GATE agraga uma série de outros serviços, que serão apresentados nesse capítulo. Os serviços do GATE são:

- **Adequação de Percepção**: converter informações entre clientes diferentes para uma exibição adequada dessas informações;
- **Serviços de Acessibilidade**: Inclui mecanismos que aprimoram o acesso do ambiente, conferindo novos meios para interação;
- **Serviços de Inter-operação**: permitir que clientes construídos em plataformas distintas acessem o servidor;
- **Serviços especializados**: outros serviços que agregam novas características ao *middleware* GATE, mas que não são imprescindíveis ao seu objetivo de alcançar todos os ambientes.

Com a definição dos graus de adequação no capítulo 4, surgem outros serviços de *middleware* para serem oferecidos. Esses serviços estão embutidos no serviço denominado de Adequação de Percepção. Cada novo serviço foi pensado para suprir um grau de adequação. Contudo, optou-se, nessa atual implementação dos serviços, não trabalhar com o 4º Grau de Adequação. Esses serviços são:

- **Conversão de dimensões**: dados de uma dimensão gráfica A são convertidos em dados de dimensão B;
- **Acessibilidade por conversação**: ferramentas de síntese e reconhecimento de fala são adicionadas ao sistema para conferir maior acessibilidade.
- **Comunicação entre idiomas**: ferramentas de tradução automática convertem texto de um idioma para o outro.
- **Interação transparente**: os quatro sub-serviços descritos acima são oferecidos de modo transparente ao usuário durante sua interação com o sistema.

Cada um desses serviços é garantido por um componente de *software* diferente. Esses componentes são formados por módulos menores especializados. Na descrição de cada

serviço é apresentada a arquitetura geral do serviço, na forma componentes de software. O desenvolvimento de cada um desses serviços gerou uma API (*Application Programing Interface*). Através do uso dessas APIs surgem as interfaces de programação entre o GATE e as aplicações que usam o GATE. Essas APIs utilizam a linguagem Java, uma vez que o próprio GATE foi desenvolvido em Java.

Vamos tratar doravante de cada um dos serviços agregados a nossa abordagem. Trataremos dos mesmos em separado começando pelo serviço de conversão de dimensões.

5.1 Conversão de dimensões

O 1º grau de adequação é também o próprio conceito de *interdimensionalidade*. As informações são convertidas graças a Lei de Correspondência da Interpercepção: cada ambiente possui uma representação textual, 2D e 3D dos seus dados. Essa representação é correspondente, pois a informação geral é definida por arquivos de meta-dados. Esses arquivos são lidos e convertidos na informação específica de um ambiente.

Quando a informação de um determinado ambiente é trocada entre aplicações clientes com interfaces de visualização de dimensões gráficas distintas, a informação será convertida. Fazendo uma analogia com uso de funções e conjuntos, se um dado é enviado por um cliente A para um cliente B, o cliente A é o conjunto domínio da função e B o contradomínio. Se A é um conjunto no $\mathbb{R}^2$ e B um conjunto no $\mathbb{R}^3$ então, ocorrerá uma transformação entre o dado enviado de A para B.

As conversões ocorrem sempre que a exibição de um dado é realizada de forma distinta em dimensões diferentes. Se por exemplo um dado sobre como um objeto O_1 é exibido no ambiente 2D for diferente da forma como o mesmo é exibido no 3D, então é necessário converter. Portanto, é vital para o processo de conversão entre as dimensões definir que dados precisam passar por uma conversão. Isso evita conversões desnecessárias e aperfeiçoa o processo.

Antes de definir que dados são convertidos, é importante definir como são representados os dados. A aplicação desse trabalho ocorre na área de sistemas distribuídos. O principal dado manipulado nesse tipo de sistema são as mensagens entre os clientes e o servidor (ou entre os clientes, mas sempre passando pelo servidor). Portanto, optou-se por definir o dado a ser convertido como sendo a mensagem. Isso também favoreceu a escolha de uma arquitetura de *middleware* orientado a mensagens para construção da nossa abordagem, o GATE.

Com o tipo de dado definido como sendo as mensagens trocadas pelo sistema, é preciso definir quais mensagens serão convertidas. A definição das mensagens que necessitam de conversão nos permite agrupá-las em classes. Inicialmente eles são agrupados em duas classes: os que precisam de conversão e os que não precisam. Sub-conjuntos podem ser definidos para melhorar a especificação do processo de conversão.

5.1.1 Definição das categorias de Mensagens

Para entender como ocorre esse processo de conversão é necessário conhecer as mensagens trocadas entre os clientes. Para tal é apresentado na Tabela 5.1, um conjunto típico

de mensagens que são trocadas entre os clientes e o servidor de um ambiente virtual. A lista de mensagens nesta tabela é provida pelo arcabouço **H-N2N**.

ID	Categoria	Nome	Descrição
H-N2N1	Controle	MJoin	Cliente solicita conexão
H-N2N2	Controle	MAnswer	Servidor aceita pedido de conexão
H-N2N3	Descrição de Objeto / Cena	MChanges	Cliente envia dados sobre mudanças
H-N2N4	Descrição de Objeto / Cena	MChanges	Servidor envia aos clientes dados sobre mudanças
H-N2N8	Dado em Tempo Real	MText	Conversação por Texto
H-N2N9	Dado em Tempo Real	MAudio	Conversação por Áudio
H-N2N10	Dado em Tempo Real	MVideo	Conversação por Vídeo
H-N2N11	Controle	MExit	Cliente avisa que está saindo

Tabela 5.1: Categorias de Mensagens

Na tabela 3.1 as mensagens são categorizadas (ou agrupadas) em mensagens de Controle, mensagens de Descrição de Objeto/Cena e mensagens de Dado em tempo Real. Nessa tabela, as mensagens são também nomeadas e receberam uma breve descrição sobre sua utilização no sistema. O ID surge como um nome alternativo para cada mensagem.

As mensagens de controle são usadas para comunicação do servidor com o cliente. Elas são enviadas pelo servidor para gerenciar o funcionamento do sistema. As mensagens de Descrição de Objeto/Cena são usadas para atualização da cena do ambiente. Elas são enviadas mediante a ocorrência de mudanças. As mensagens de Dado em Tempo Real são usadas para transmissão de informações em tempo real e comunicação (também em tempo real). Elas são envidas quando o tipo de dado específico da mensagem (texto, áudio ou vídeo) é enviado por algum cliente.

A lógica de funcionamento de uma aplicação distribuída é ditada pelo uso das mensagens de Controle. Assim que o usuário entra no sistema pela primeira vez, o cliente envia a mensagem *MJoin* ao servidor. Nesta mensagem é informado o *nick* (apelido) do usuário e o seu avatar. Em um ambiente que não usa uma representação gráfica 2D ou 3D, o avatar será um texto com o próprio *Nick*. O servidor responde aceitando a conexão com *MAnswer*. Quando um usuário deixa o ambiente, o cliente envia uma mensagem de *MExit*.

As mensagens de Dado em tempo real fornecem ações ao usuário. A ação mais simples fornecida aos usuários (por uma aplicação cliente) é conversação (ferramenta de batepapo). Essa conversação pode ser por texto (que faria o cliente enviar uma *MText*) ou áudio (que faria o cliente enviar uma *MAudio*) . Existe ainda a possibilidade de a conservação ser por vídeo (uma conversação gestual), fazendo o cliente enviar uma *MVideo*. As mensagens da categoria Dado em Tempo Real também podem ser usadas para transmissão ou execução de texto, áudio ou vídeo na cena do ambiente. As demais ações disponíveis

ao usuário estão ligadas a sua interação com o espaço virtual do ambiente. As interações do usuário com o ambiente geram mudanças dinâmicas na cena que está sendo exibida no momento. Essas mudanças são sinalizadas pelo cliente com a mensagem *MChanges*. Ao receber uma mensagem desse tipo, o servidor envia outra versão dela para os demais clientes (todos os clientes com exceção daquele que enviou a *MChanges*). As mensagens agrupadas na categoria de mensagens de Descrição de Objeto/Cena é também a classe das mensagens que precisam de conversão. As mensagens de controle são idênticas para qualquer interface usada pelo ambiente. O mesmo vale para as mensagens de Dado em tempo real.

As mensagens de Descrição de Objeto/Cena descritas na tabela 3.1 são muito abrangentes. Especializações desse tipo de mensagens são importantes para definição de interações possíveis em um ambiente virtual. A quantidade de ações permitidas em um ambiente é variável e dependente dos objetivos de cada ambiente. Por isso, além das mensagens já fornecidas pelo arcabouço **H-N2N** (que permite a construção do sistema distribuído), outras mensagens devem ser criadas de acordo com as necessidades de cada ambiente (ou aplicação). As mensagens da Tabela 3.1 não permitem, por exemplo, que o usuário realize uma ação como "mover-se". Por isso, na Tabela 5.2 é apresentado um novo conjunto de mensagens. Essas mensagens foram providas pela ferramenta de criação de ambientes virtuais PercepCom [Dantas et al. 2005].

As mensagens de Descrição de Objeto/Cena descritas nessa tabela apresentam novas ações permitidas ao usuário. Elas representam um conjunto básico de mensagens para ambientes virtuais. Num outro ambiente virtual pode existir outras mensagens além dessas. Contudo, procuramos restringir o conjunto de mensagens por conveniência do nosso estudo.

A mensagem MMover delimita as ações de movimento dos avatares. São usadas para realizar as transformações afins necessárias para representação do movimento dos avatares. A mensagem MMudarSala é enviada pelo cliente pedindo ao servidor que o mude de sala dentro do ambiente. Essa mensagem pode ser vista como uma especialização da *MMover*. Contudo, a ação de "mudar de sala" envolve um tempo de carregamento dos dados da nova sala. Ela também pode envolver a mudança de grupos dentro de um sistema, uma vez que o **H-N2N** prevê a criação de grupos.

A mensagem *MAddUsuario* é usada para a adição de novos usuários depois que ambiente já está em execução. Isso significa dizer que ela adiciona os avatares de novos usuários à cena de um cliente que já estava conectado. Essa mensagem pode ser vista como sendo de controle, mas ela está ligada diretamente a mudanças na cena. Além disso, ela envolve a adição de novos objetos dinâmicos (os avatares) à cena.

Além de novas ações, as mensagens da Tabela 5.2 também adicionam novas possibilidades de controle. A mensagem *MNickExistente* é enviada pelo servidor para o cliente no caso do nick enviado pelo cliente já estiver em uso. A mensagem *MMundo* é a responsável pelo armazenamento de uma cópia do mundo virtual na aplicação cliente. O servidor envia essa mensagem logo após a entrada e aceitação de um novo cliente. É importante salientar que essa cópia do ambiente também trás uma lista dos usuários que estavam conectados no ambiente naquele instante. No caso do uso da mensagem *MAddUsuario*, após uma *MMundo*, um novo usuário seria adicionado à lista de usuários. A existência dessa

ID	Categoria	Nome	Descrição
PER1	Descrição de Objeto /Cena	MAddUsuario	Servidor adiciona avatares de novos clientes
PER2	Controle	MNickExistente	Servidor informa que um Nick já esta em uso
PER4	Controle	MMundo	Servidor envia uma cópia do ambiente para o cliente
PER5	Descrição de Objeto /Cena	MMudarSala	Cliente troca de sala no ambiente
PER6	Descrição de Objeto /Cena	MMover	Cliente se move com seu avatar

Tabela 5.2: Mensagens de comunicação do Percepcon

lista também possibilita ao usuário a ação de "listar os usuários" em um dado momento.

As mensagens de controle novas não precisam de conversão. Na verdade, as mensagens de controle estão no grupo das mensagens que não precisam de conversão, assim como as mensagens de Dado em Tempo Real. Já as mensagens de Descrição de Objeto/Cena são o foco do processo de conversão. As especializações mostradas na tabela 5.2 apresentam meios para discutirmos sobre o processo de conversão entre as dimensões.

5.1.2 Convertendo mensagens de movimento

As mensagens de *MMover* são compostas pela tupla $< origem, vetordepontos >$. A origem representa a dimensão geométrica da interface do cliente que enviou a mensagem. O vetor de pontos é usado para gerar um movimento através de uma função de interpolação entre os pontos.

Se os ambientes possuem dimensões geométricas diferentes é natural que eles representem de forma distinta seus pontos. Para realizar a conversão do movimento realizado pelos avatares é necessário definir como os pontos são representados em cada ambiente. Num ambiente 3D os pontos são formadas por vetores com 3 coordenadas (que representam altura, largura e profundidade). No ambiente 2D uma dessas coordenadas não é utilizada. A coordenada não utilizada irá depender do ambiente. Um ambiente desenhado com projeção iria descartar a altura, por exemplo. No ambiente 1D, a representação dos dados é uma abstração descritiva dos demais.

A coordenada removida do 2D depende do design do ambiente. Por exemplo, num ambiente com visão isométrica, existe apenas largura e profundidade. Num ambiente com visão de câmera em 3ª pessoa existe largura e altura. A não existência de uma coordenada no 2D nos leva a uma hipótese para conversão.

> **Hipótese 5.1**: Ao converter um ponto do 3D para o 2D, uma coordenada deve ser descartada. A coordenada descartada deve ser definida pela função que realiza essa conversão.

Partindo da Hipótese 5.1, converter um dado do 3D para o 2D é definir uma transformação $T : \mathbb{R}^3 \rightarrow \mathbb{R}^2$ que leva um ponto do 3D do ambiente em um ponto do 2D removendo uma das coordenadas. Se um ponto no 3D é representado pelo vetor $v(x,y,z)$ e um ponto no 2D pelo vetor $u(a,b)$, a transformação sugerida pode ser definida pela Equação 5.1. Na Equação 5.2 temos a transformação no outro sentido: 2D para o 3D.

$$T(a,b) = (x,y). \tag{5.1}$$

$$T(x,y,z) = (a,b,k). \tag{5.2}$$

Na Equação 5.2 temos a introdução de uma constante k na 3ª posição do vetor do 3D. Isso ocorre, pois não existe um valor no 2D para essa coordenada. Se essa constante será sempre adicionada no processo então, podemos usar o conceito de vetor homogêneo [Hartley & Zisserman 2003]. Segundo esse conceito, um vetor do $\mathbb{R}^2$ pode ser representado por um vetor do $\mathbb{R}^3$ que possui uma coordenada constante não nula. O valor ideal para essa constante é 1 por ele ser o elemento neutro da multiplicação. Assim, ele pode ser removido do vetor sem mudar o valor das demais coordenadas. O uso de vetores homogêneos nos permite reescrever a Equação 5.1 como a Equação 5.3 .

$$T(a,b,1) = (x,y,k). \tag{5.3}$$

A Equação 5.3 mostra uma forma mais geral de conversão de pontos do 2D para o 3D. A transformação da Equação 5.3 não é completamente geral o que nos leva a uma nova hipótese que busca generalizar essa transformação.

> **Hipótese 5.2**: Se for assumido que todos os pontos do 2D podem ser diretamente mapeados no 3D então isso nos leva a equação geral de transformação (Equação 5.4).

$$T(x,y,1) = (x,y,k). \tag{5.4}$$

Pela Equação 5.4 é possível gerar um ponto para o ambiente 3D, preservando o valor de duas coordenadas e gerando um valor para uma terceira. Essa equação apresenta um modelo simples e também vago de transformação. Esse modelo pode ser adaptado para o ambiente em que será usado. Por exemplo, num ambiente em que o 2D é construído pelo uso de *tiles*, deveria está representado na equação, uma constante referente proporção entre o *tile* e uma região equivalente no 3D. Outra relação possível é razão entre a entre a área do 2D e a área do 3D, caso essas áreas sejam diferentes. A Equação 5.5 mostra uma transformação entre pontos do 2D para o 3D, onde as áreas não são proporcionais e no 2D a área é dividida em *tiles* de tamanho constante.

$$T(x,y,1) = ((x+l/2)/r,(y+l/2)/r,k). \tag{5.5}$$

A Equação 5.5 mostra que existem formas diferentes de calcular a transformação dos pontos entre as dimensões geométricas. A constante l representa o lado do *tile* e r é a razão entre a área do ambiente 2D e a área do ambiente 3D. Apesar da adição de novas

constantes, à Equação 5.5 poder ser reduzida a equação 5.4. A partir da variações da Equação 5.5(como a equação 5.5) é possível criar conversões entre quaisquer modelos de ambientes 2D e 3D. Para conversão entre 1D e outra dimensão, novas hipóteses devem ser levantadas.

> **Hipótese 5.3**: Se num ambiente 1D a posição é uma abstração descritiva representar a posição com um vetor é uma tarefa desnecessária.

Por definição espacial, um ambiente 1D deveria ser formado por vetores com uma coordenada (vetores do $\mathbb{R}^1$). Contudo, a definição empregada nesse trabalho para ambiente 1D é a de uma ambiente cuja representação é descritiva (textual). Assim, suas posições não são vetores, mas palavras ou frases. Quando um usuário deseja ir a um determinado local numa sala de uma ambiente 1D ele usa um comando, uma palavras reservada do vocabulário do ambiente. Esse vocabulário é composto por comandos (palavras reservadas) que representam as ações possíveis no ambiente. Assim as interações dos usuários do 1D são realizadas pelo emprego desses comandos.

Um comando poderia permitir que o usuário fosse de um extremo a outra de uma sala, ou mesmo abrir uma porta e ir para outra sala. Nesse tipo de comando um deslocamento, mesmo que abstrato, ocorre entre dois pontos. Esses pontos deveriam ter alguma representação para que seja possível realizar a conversão entre o movimento do 1D e o movimento do 2D, por exemplo.

A solução proposta é empregar pontos de interesse no ambiente 1D. Comandos que envolvam deslocamento iram fazer referência a esses pontos. Esses pontos são os únicos lugares espaciais acessíveis no 1D. Para converter o movimento gerado no 1D para outro ambiente, deve ser armazenada as coordenadas desses pontos de interesse. Um vetor 1D homogêneo como com o formato $v(x,1,1)$ poderia ser usado. Contudo, para que seja possível gerar um movimento fiel entre o 1D e o 2D por exemplo, uma segunda coordenada deve ter valor não fixo. Por isso, os vetores do 1D devem ser representados por vetores homogêneos do 2D.

Quando um deslocamento é realizado pelo 1D é armazenada a informação sobre os dois pontos de interesse contidos nesse deslocamento. Através do emprego de um algoritmo de interpolação é criada a nuvem de pontos entre os dois pontos. Essa nuvem é usada para gerar um movimento suave entre os dois pontos equivalente no 2D. Se essa nuvem não fosse gerada, a movimento de um usuário do 1D seria percebido no 2D como tele-transporte entre dois pontos. Uma vez que os pontos do 1D são representados por vetores 2D, converte-los para o 3D segue o processo descrito anteriormente para converter 2D para 3D. O algoritmo geral para conversão de movimento é representado no diagrama de atividade da Figura 5.1.

As atividades descritas no diagrama da Figura 5.1 sintetizam o processo de conversão de movimento descrito até então. A interface de origem usada na verificação inicial é uma informação presente na própria mensagem de movimento. Já o destino é definido pelo Portal que recebe a mensagem: o destino será a interface do cliente ligado a esse Portal. Caso a interface de origem seja 1D, será necessário gerar pontos para tornar o movimento suave entre duas posições. Dessa forma, a conversão entre as dimensões está completamente definida. Portanto, iremos tratar agora da conversão de avatares.

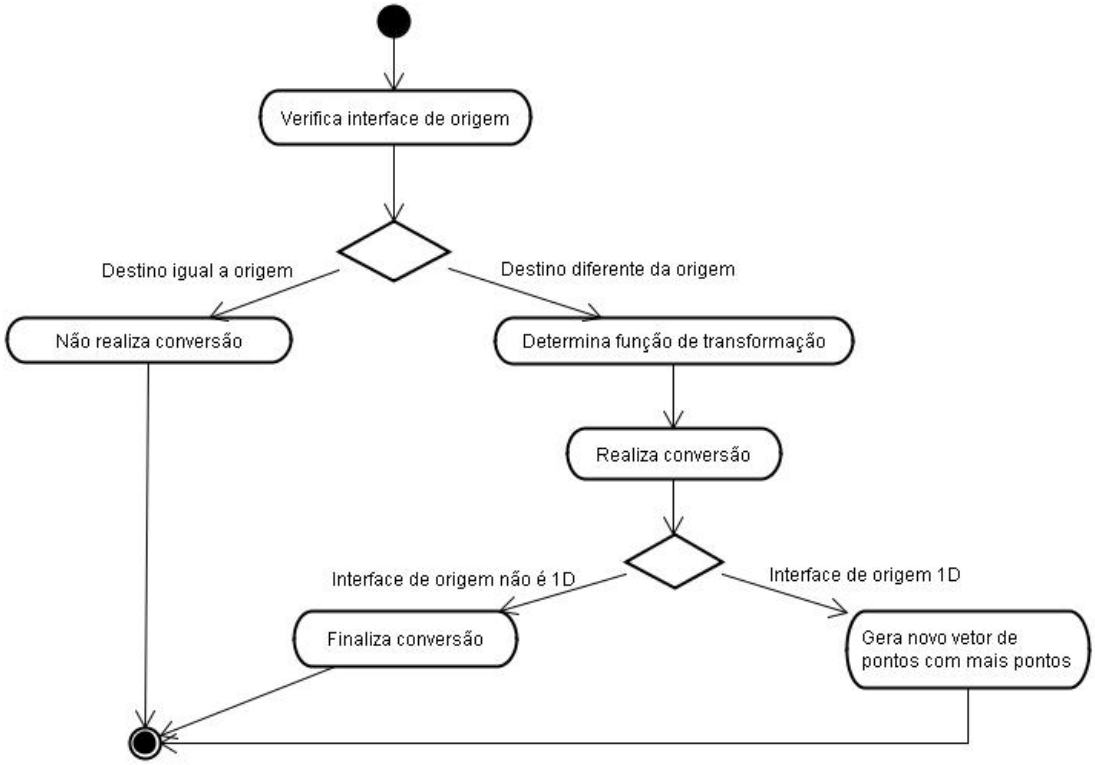

Figura 5.1: Conversão de Movimento

5.1.3 Conversão de avatares

Para adequação de percepção é necessário garantir que cada interface de visualização possua um modelo de avatar condizente com a sua dimensão. Assim, como foi necessário converter o deslocaments realizado esses avatares, surge a necessidade de converter a representação dos mesmos.

As mensagens para adição de avatares são compostas de uma tupla <*apelido, tipo, corpo, posição, orientação*>. O apelido é o nome dado pelo usuário ao seu avatar. O tipo classifica os avatares em grupos, e é importante para delimitar o avatar no momento da conversão. O corpo define a estrutura física de um avatar. A posição representa o ponto espacial onde o avatar estará no momento em que for adicionado ao ambiente. A orientação delimita o ângulo de orientação para onde o avatar se desloca.

O tipo de um avatar é uma característica que independe da interface de visualização. O tipo é usado como uma referência para determinar o corpo de um avatar no momento da conversão. Por exemplo, o tipo criança definiria um modelo com características do corpo de uma criança. O corpo por sua vez é dependente da interface. O corpo num ambiente 2D é representado por um vetor de imagens, geralmente definido na literatura como *Sprite* [Jones 2000]. O corpo no ambiente 3D é representado por um modelo 3D. No ambiente 1D o corpo é representado por uma descrição textual. A característica posição também depende da interface de visualização. Como discutido na seção anterior, a representação de posição depende da dimensão geométrica do ambiente. A característica

orientação é imprescindível no ambiente 3D, mas é desnecessária as demais.

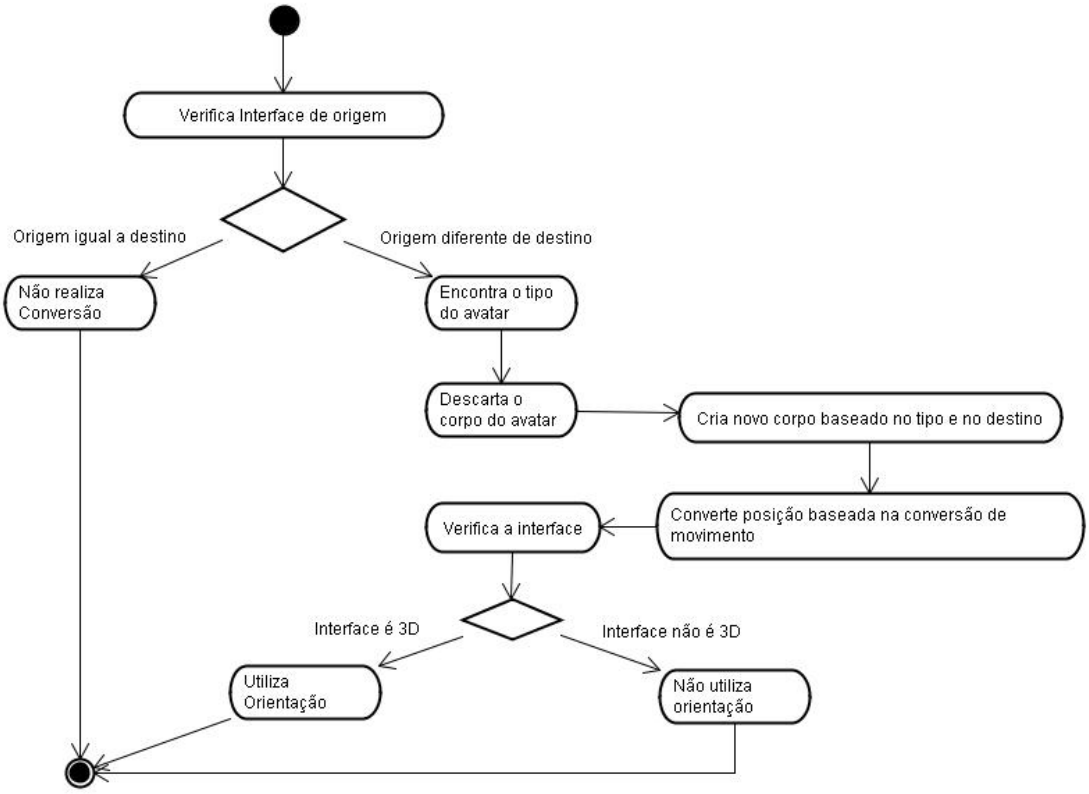

Figura 5.2: Processo de conversão de avatares.

Quando a mensagem de adição de avatares já está em algum Portal, esse Portal determina a interface de origem a partir do dado contido no campo posição. Como o ambiente 1D e 2D usam a mesma representação da posição, o corpo será usado para definir quando o cliente é 1D ou 2D. O cliente acoplado a esse Portal delimita a interface de destino. Deste ponto em diante o Portal segue o algoritmo descrito no diagrama de atividade da Figura 5.2 para conversão dos avatares.

Como demonstra o diagrama da Figura 5.2 , as informações contidas na mensagem de adição de avatares são utilizadas para criar um avatar antes de sua adição (atividade de criação de um novo corpo). Cada interface de visualização tem um tipo específico de avatar. Todos estes avatares descendem de um avatar padrão. Definimos um avatar padrão chamada apenas de Avatar e depois criamos as especializações dele: Avatar 1D, Avatar 2D e Avatar 3D. Quando o Portal recebe uma mensagem para adição de avatares ele segue o algoritmo descrito no diagrama da Figura 5.2 . Através deste algoritmo o Portal cria um novo avatar, adequado à interface de visualização a qual ele está acoplado.

Depois de criado o avatar novo, ainda é feito o ajuste de posição. Caso o avatar seja direcionado a um ambiente 3D, uma nova informação sobre orientação é referenciada ao avatar. Dependendo da forma como o ambiente 2D foi construído também é possível que o mesmo use alguma informação sobre orientação. Por isso, uma modificação na atividade final pode ser feita para melhor adequação do ambiente.

5.1.4 Considerações sobre o serviço de conversão

A descrição do processo de conversão deixa claro que as funções usadas para conversão não são definitivas. Foram apresentadas formas gerais para conversão. Seguindo essas formas gerais também foram mostradas especializações, condizentes com certos modelos de ambientes. Para criar um serviço robusto, várias especializações devem ser oferecidas. Cabe ao cliente desse serviço escolher qual especialização usar para atender suas necessidades.

O trabalho de [Rolim et al. 2007] apresenta operadores para generalização cartográfica para ambientes 3D. A operação de conversão de 3D para 2D pode ser vista como uma generalização do operador *Colapso* apresentado no trabalho citado. Da mesma forma, a conversão 2D para 3D pode ser visto como uma generalização do operador Exageração. Esses operadores servem como base para definição de especializações futuras. Não faz parte dos objetivos desse trabalho apresentar todas as especializações possíveis, deixando essa tarefa a cargo de trabalhos futuros. Seguimos agora com a descrição dos demais serviços.

5.2 Serviços de Acessibilidade

Os serviços agrupados nessa categoria têm o intuito de fornecer acessibilidade ao ambiente. Esses serviços facilitam a interação do usuário com o ambiente. Eles também buscam oferecer novos meios de realizar as interações possíveis no ambiente. Com isso é garantido um acesso que visa minimizar ao máximo as restrições de uso. Contudo, esses serviços ainda não são capazes de atender certos perfis de usuários. Por exemplo, as ferramentas que serão apresentadas não permitem a integração de usuários com deficiência auditiva, ou motora. Esses aspectos serão tratados em trabalhos futuros.

5.2.1 Acessibilidade por conversação

O 2º grau de adequação prevê o uso de ferramentas de síntese e reconhecimento de fala. Uma ferramenta de síntese de fala produz fala através de entradas textuais. Essas entradas textuais podem estar previamente armazenadas em um arquivo. Também é possível estabelecer uma síntese dinâmica, onde o arquivo de texto a ser sintetizado é recebido em tempo real.

No reconhecimento, um padrão ou comando é reconhecido. Alguma ação associada a esse reconhecimento pode ser disparada. Portanto, numa ferramenta de reconhecimento de fala os comandos de fala são as entradas. Para que um comando seja reconhecido ele deve ser associado ao vocabulário do reconhecedor. No vocabulário cada comando reconhecível é associado a uma ação. Essa ação é disparada em resposta ao reconhecimento do comando.

Quando descrevemos os ambientes do tipo 1D descrevemos os mesmos como sendo formados por simples descrição textual. Os dados usados na representação textual de um ambiente 1D também podem ser usados para alimentar as ferramentas de síntese e reconhecimento de fala. Os textos usados para representação do ambiente (incluindo usuários,

objetos e o ambiente em si) são usados como entrada para o sintetizador. Os comandos usados para representar as ações possíveis dentro do ambiente são usados para definir os mesmos comandos que formam o vocabulário do reconhecedor. O funcionamento desse serviço é descrito na arquitetura da Figura 5.3.

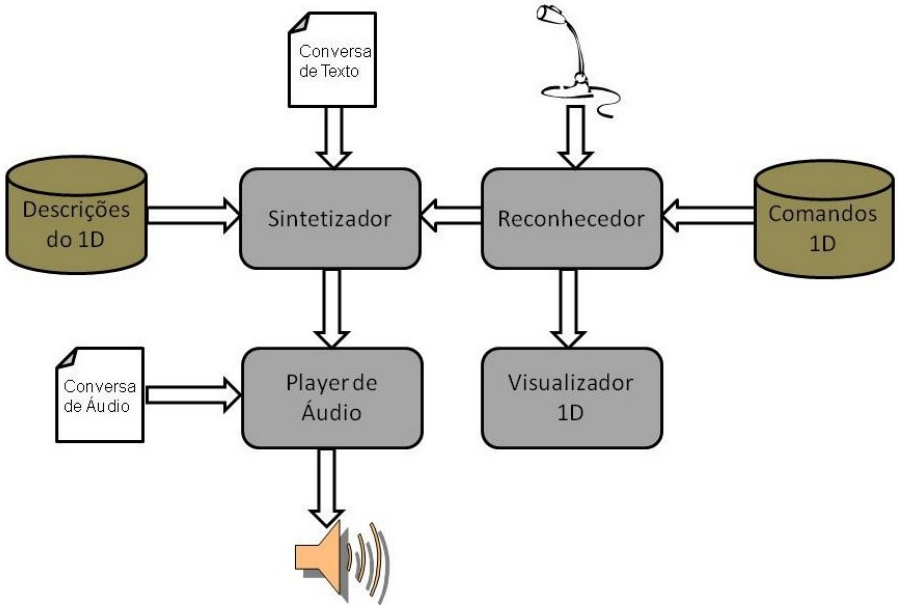

Figura 5.3: Arquitetura de funcionamento do serviço de acessibilidade por conversação

Na arquitetura da Figura 5.3 temos um banco com descrições do 1D servindo como entrada para o sintetizador. Arquivos com conversa de texto também são usados como entrada dessa ferramenta. Os ambientes aqui descritos prevêem o uso de conversação por áudio. Esse tipo de conversação não necessita de síntese, podendo ser transmitida da forma que é exibida. Já a conversação por texto deve ser sintetizada para o uso do sistema por uma pessoa com debilidade visual. Assim, a saída do sintetizador vai para um *player* de áudio. Esse player recebe diretamente as conversas de áudio.

O reconhecedor, segunda a Figura 5.3, possui como entradas os comandos do 1D e comandos recebidos por microfone. O reconhecedor compara os dois comandos e se forem iguais dispara uma ação como saída. Essa ação de saída tem um resultado que deve ser exibido na interface de visualização do 1D. Ao ser exibido, o módulo de controle do 1D envia a mensagem referente a ação. Essa mensagem será recebida pelos demais clientes. O resultado da ação deve também ser sintetizado para interação por uma pessoa com debilidade visual.

As ferramentas associadas a este serviço são ligadas ao uso da fala como meio de interação. Além disso, seu intuito é fornecer maior acessibilidade aos ambientes. Por isso demos ao serviço o nome de Acessibilidade por conversação. Como sua descrição mostra, esse serviço funciona como uma camada de *software* que executa sobre o aplicativo do ambiente 1D.

5.2.2 Interação transparente

Quando um usuário interage através de ambiente virtual ele possui um conjunto ações. Essas ações foram chamadas anteriormente nesse trabalho de ações de interação. São tidas como ações de interação porque delimitam as formas possíveis de que o usuário possui para interagir.

A interação pode ser feita com o ambiente ou com outros usuários. Isso agrupa as ações em duas categorias: ações de interação com o ambiente e ações de interação com o usuário. Entre as interações com o ambiente podemos listar: andar, observar o ambiente, abrir uma porta, correr, etc. Já nas ações de interação com usuários podemos listar: conversar, observar pessoas, abraçar, etc.

O que chamamos de interação transparente vem da possibilidade de uma determinada ação ser usada em qualquer ambiente independente da dimensão geométrica usada pelo mesmo. Assim se uma determinada ação é criada para a interface 3D, ela deve ser habilitada na interface 2D e 1D. Isso faz que criação de uma ação seja seguida da definição de uma função que transforme essa ação em uma ação de outra dimensão. Esse modelo conceitual é mostrado na arquitetura da Figura 5.4.

De acordo com a Figura 5.4 um módulo de transformação recebe como entrada um vocabulário de ações. Esse vocabulário pode ser um arquivo de meta-dados com a descrição de uma ação. Um banco funções de transformação tem é dado como entrada do módulo de transformação. A saída são banco de ações definidas para cada uma das interfaces.

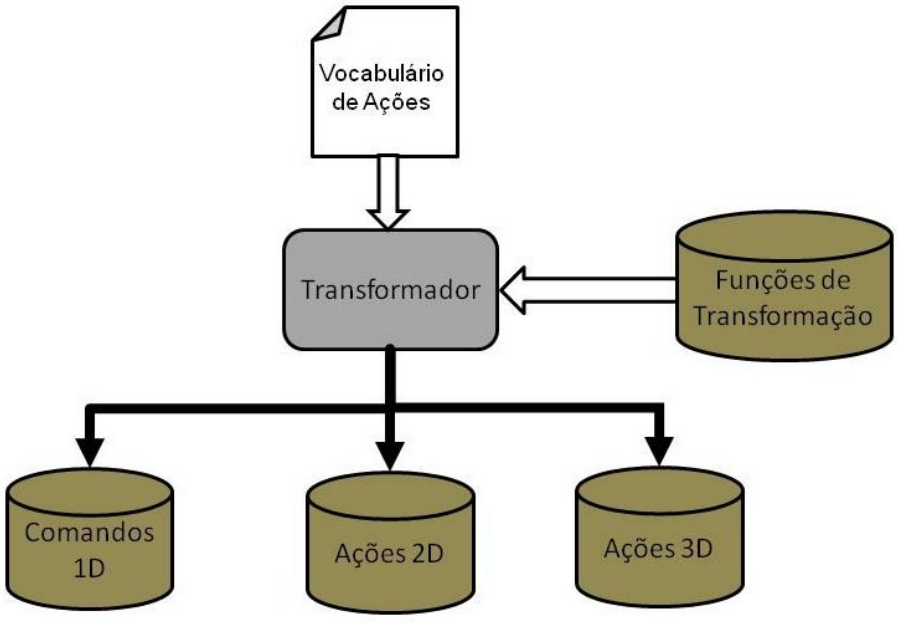

Figura 5.4: Arquitetura do serviço de interação transparente.

Por exemplo, a ação de "andar" foi definida para existir nos ambientes citados nesse trabalho. Por isso, foi necessário definir uma função que converte a tarefa de andar em cada um dos ambientes. Esse mesmo processo deve ser repetido para cada ação. Por

isso, o serviço de interação transparente cria um vocabulário de ações de interação. Essas ações são transformadas para se adequar a cada interface. Essa adequação é feita de uma forma que faz o acesso ser transparente, pois não fornece meios que permitam distinguir de qual interface os outros usuários estão interagindo.

5.2.3 Comunicação entre idiomas

O 3º grau de adequação da interpercepção diz respeito à interação dos usuários através da comunicação por bate-papo (**chat**). Independente de ser realizado por digitação de texto ou comunicação por voz, esse grau de adequação deve realizar a conversão da comunicação trocada entre os usuários. A conversão feita por este grau é entre idiomas, portanto prevê o uso de uma ferramenta de tradução.

A tradução simultânea durante uma conversação por áudio é algo custoso. Não faz parte do objetivo do nosso *middleware* nesse momento realizar esse tipo de tradução. Portanto, decidimos atacar o problema da tradução por comunicação textual.

Existem muitos tradutores automáticos atualmente na *Internet*. Por muitos anos essas ferramentas foram notórias pela baixa qualidade de sua tradução. Essa baixa qualidade estava ligada a uma tradução literal e puramente sintática. A qualidade de uma tradução está no seu contexto semântico, algo difícil de conseguir em um processo automatizado de tradução.

Pare textos formais esse tipo de tradução não é útil. Contudo, o texto que estamos tratando aqui é o texto de um bate-papo, que é um texto informal. Por isso, o uso de uma ferramenta de tradução automática já resolveria parte do 3º grau de adequação. Existem muitas dessas ferramentas na *Internet*. Por isso, torna-se mais interessante usar uma dessas ferramentas ao invés de desenvolver uma nova.

A ferramenta de tradução do Google, hoje é oferecida como serviço. Essa ferramenta pode ser acoplada como serviço a páginas da *Internet*. Optamos por incorporar o mesmo como um serviço ao nosso *middleware*.

Outra tarefa envolvida com a conversão de idiomas é a internacionalização das interfaces. Através de um arquivo de meta-dados são definidos os textos que compõem a interface gráfica com o usuário. Esses textos são associados a sua tradução para diversos idiomas. Quando o usuário escolhe o cliente que usará parar acessar o ambiente, ele também escolha a língua que será utilizada.

5.3 Serviços de Inter-operação

Para cada novo cliente que possa ser interligado ao sistema, um novo Portal pode ser criado. Isso garantido pelo estabelecimento do **GATE** como um *middleware* orientado a mensagens (MOM). Como dito anteriormente no capítulo 4, o uso de MOM também fornece uma arquitetura fracamente acoplada, facilitando, portanto a adição de novos componentes Portais. Para interligar esses clientes devem ser definidos mecanismos que permitam a comunicação deles sem restrições.

No início do capítulo 4, nós falamos das camadas de diversidade existentes em um sistema distribuído. Até o presente momento apresentamos serviços para solucionar a

diversidade de pessoas. Agora colocamos os serviços de inter-operação que solucionam a diversidade de dispositivos e de redes.

5.3.1 Inter-operação de clientes

A interpercepção, como originalmente proposta, trabalha na camada de pessoas. Ela proporciona a interligação de aplicações com interfaces de visualização distintas para atender a diversidade de usuários. Devido ao fato de que pretendemos atender a uma diversidade de requisitos, faz-se necessária uma modificação em vários componentes do **H-N2N**. Por isso decidimos realizar uma adaptação do **H-N2N**. Para entender esse processo de adaptação vejamos como era a estrutura do **H-N2N**, do ponto de vista do seu diagrama de classes. O mesmo é apresentado na Figura 5.5.

Nessa figura temos a classe central e abstrata denominada CommunicationLink. Essa classe possui duas subclasses *LinkUDP* e *LinkTCP*. Através das subclasses são criadas as conexões entre os clientes. Após uma conexão ser estabelecida, ela é observada por uma instância da classe *LinkObserver*. Essa classe notifica sobre o recebimento de novos pacotes. Os pacotes são criados como instâncias da classe *Package*. O método *sendMessage* é usado para o envio dos pacotes. Esse método é definido nas classes *LinkUDP* e *LinkTCP*.

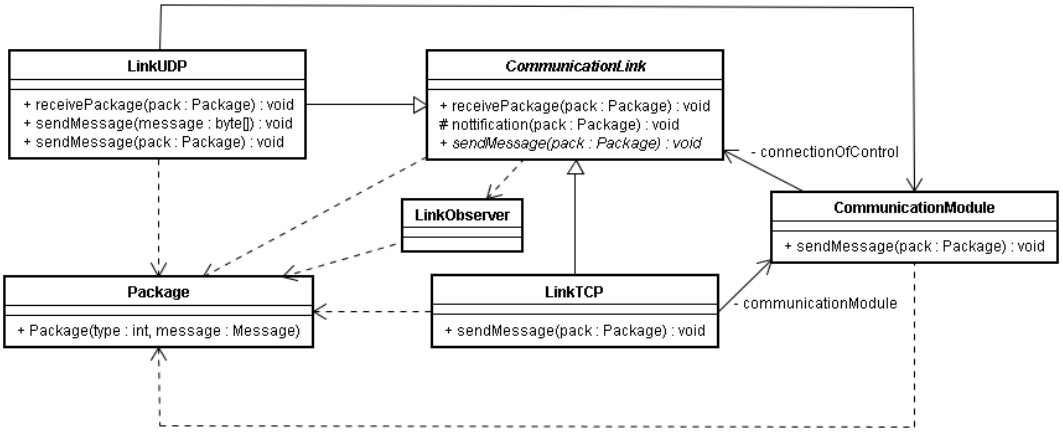

Figura 5.5: Diagrama de Classes do H-N2N

A classe *CommunicationModule*, que também aparece no diagrama da Figura 5.5 é responsável por reunir todas as conexões criadas. Ele possui um vetor de conexões e atua no controle das mesmas. Quando uma conexão é criada, a própria conexão se registra ao *CommunicationModule*. O método *sendMessage* dessa classe é o responsável por enviar pacotes dos clientes para o servidor.

O **H-N2N** envia e recebe objetos serializáveis e com isso a única possibilidade de comunicação entre aplicações são as que são construídas utilizando a linguagem Java. Para permitir a integração dos diferentes dispositivos, é necessário construir um serviço *middleware* que permita a comunicação inter-operável entre os aplicativos clientes. A diferença de dispositivos está ligada à diferença de linguagens disponíveis para o desenvol-

vimento das aplicações. É necessário, portanto alterar o modo de comunicação utilizado na implementação do **H-N2N**.

5.3.2 Adaptação do H-N2N

Para que possamos construir aplicações com hardwares e softwares distintos, foi definido um protocolo binário de comunicação. Este protocolo é capaz de comunicar os diferentes dispositivos e linguagens. Mesmo com essa definição de protocolo, temos outros problemas relacionados com o formato de armazenamento e transmissão de dados binários. Neste caso, o formato *BigEndiam* e *LittleEndiam*.

Diante destas questões e novas funções, decidimos fazer uma adaptação do **H-N2N**, dando origem a uma nova versão do mesmo chamada de **H-N2NInt**. O seu nome é uma referencia ao fato dele permitir a inter-operação entre linguagens e, conseqüentemente, entre aplicações clientes e os dispositivos usados para executar essas aplicações. Essa inter-operação é garantida pela forma como as mensagens são construídas e tratadas pelo servidor.

5.3.3 Construção das mensagens

Definindo apenas o formato dos dados binários não é suficiente para garantirmos a interoperabilidade. Muitas linguagens enviam dados binários pela rede de forma diferente, como por exemplo, *Strings*. *Strings* na linguagem C são apenas um vetor de caracteres enquanto que na linguagem Java são objetos contendo, além da *string*, informações sobre a codificação utilizada. Isso faz com que uma transmissão de uma *String* seja correta com programas desenvolvidos pela mesma linguagem, mas incorreta quando utilizamos linguagens distintas.

Para resolver esse problema, criamos um componente que realiza a codificação das informações que serão trocadas entre os programas. Essa codificação representa todos os dados em binário e armazena os mesmos em um *buffer*. Esse *buffer* é transmitido pela rede. Do outro lado, um componente recebe e decodifica o *buffer*. Nestes componentes as informações são armazenadas de modo único. As implementações desses componentes tem que seguir um protocolo que será especificado mais a frente.

Esse processo de codificar e depois decodificar é baseado no padrão de projeto remoto *Marshaller* [Volter et al. 2005]. Segundo esse padrão, as mensagens são convertidas em um stream de bytes no lado do cliente. As mensagens são enviadas pela rede. Ao chegar ao servidor às mensagens são traduzidas e transformadas na informação apropriada. O processo de converter a mensagem é chamado de *Marshalling*. O processo de traduzir a mensagem ao seu formato original é chamado de *Unmarshalling*. Baseado no conceito desse padrão a estrutura do **H-N2N** foi refeita, gerando o **H-N2NInt**. O diagrama de classe do **H-N2NInt** é apresentado na Figura 5.6.

A principal diferença entre o modelo conceitual do **H-N2N** para o **H-N2Nint** é a ausência do *Package*. Ele foi quebrado em duas novas classes (apresentadas na figura 5.6 com a cor mais escura). As mensagens no arcabouço adaptado são construídas utilizando o componente de *software* que realiza o *Marshalling* (*BufferWrite*) e *Unmarshalling* (*Buf-*

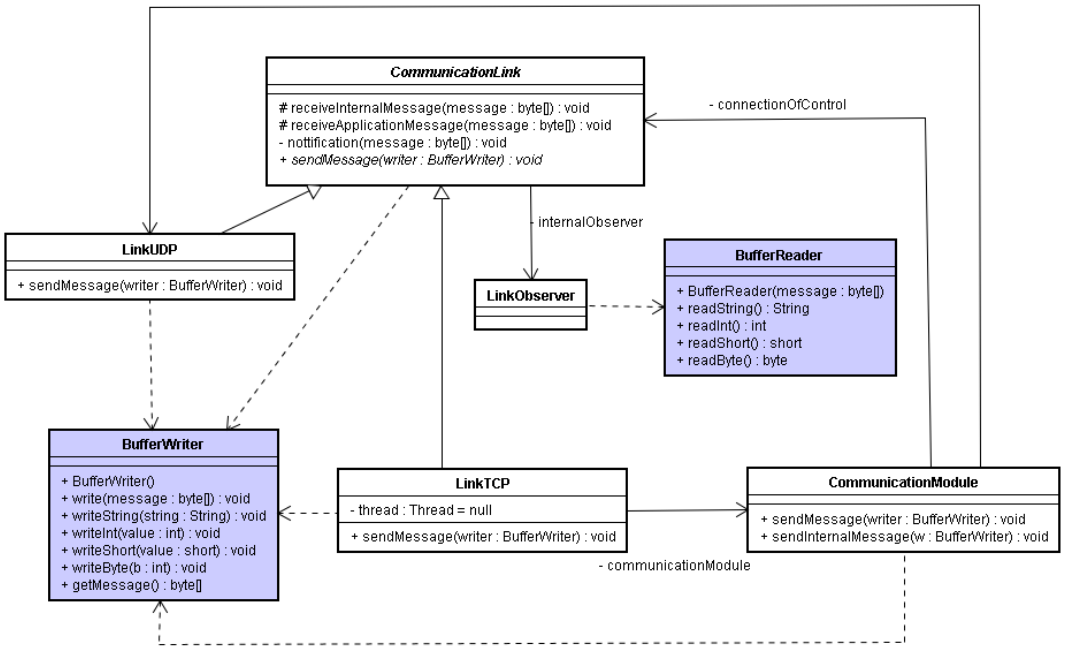

Figura 5.6: Diagrama de classes do H-N2NInt

ferReader) das informações binárias. Com isso, não temos ambigüidade da forma de como os dados são enviados/recebidos pelas aplicações. Esses componentes ficam responsáveis por realizar as transformações necessárias para que a informação binária seja corretamente interpretada. Nessa versão, apenas os seguinte dados binários foram utilizados:

- byte - 8bits
- short - 16bits
- int - 32bits
- string - vetor de caracteres

Considerando os tipos de dados acima, para armazenar as informações, propomos realizar algumas transformações nos dados para garantir que a informação constante deles seja escrita/lida de maneira unicamente interpretável. O programador receberá as informações no tipo de dado da linguagem que está utilizando, sendo desnecessário realizar modificações nas estruturas de dados. As mensagens são escritas da seguinte forma:

- byte - Não sofre nenhum tipo de transformação
- short e int - São transmitidos utilizando o formato BigEndiam
- string - É transmitido primeiramente o tamanho da string no formato BigEndiam e depois transmitido somente os caracteres que fazem parte da string

As mensagens são lidas de forma análoga a escrita. Para que uma mensagem seja enviada e recebida tanto pela aplicação quanto pelo servidor, ambos somente têm acesso a esse componente de *software*. Com isso, garantimos que a mensagem seja sempre transmitida no formato definido acima.

Fazendo dessa forma, a mudança na maneira de como os dados é transmitida pela aplicação se torna transparente. Isso permite melhorar ou modificar a forma de como os dados são transmitidos sem modificar as aplicações. Nesses componentes podemos adicionar mecanismos de criptografia, para uma transmissão mais segura das informações. A criptografia é possível se levarmos em conta que a mesma será conhecida em ambos os lados e implementados de maneira similar em ambas as linguagens.

5.3.4 Inter-operação de redes

As aplicações criadas com o arcabouço **H-N2N** permitem conexões UDP e TCP, como mostrado pelo diagrama de classe do arcabouço. O servidor também não faz distinção quanto ao uso de redes com ou sem fio. É possível, por exemplo, criar um servidor em dispositivo móvel. Contudo, não é possível criar conexões via *Bluetooth*.

Na revisão do arcabouço foi estabelecido num novo tipo de conexão *LinkBluetooth*. Assim como os *LinkTCP* e *LinkUDP* definidos anteriormente, esta estrutura é derivada de *CommunicationLink*. Numa aplicação com uso de rede *Bluetooth* e, são criadas conexões apenas do tipo *LinkBluetooth*. Os demais componentes dessa estrutura continuam funcionando da forma descrita anteriormente.

O *LinkBluetooth* tem limitações inerente a rede *Bluetooth*: alcance e quantidade limitada de clientes. O alcance é uma barreira que varia de acordo com a tecnologia do dispositivo e não está entre nossos objetivos atacá-la. Já a quantidade pode ser superada com o uso de hierarquia de servidores. Normalmente, apenas 7 cliente são suportados numa rede *Bluetooth*. Contudo, um dos 7 cliente pode ser eleito servidor e e fornecer serviço para outros 7 novos clientes.

Essa estrutura é permitida pelo arcabouço **H-N2NInt**. Portanto é possível, usando o mesmo, criar uma aplicação *Bluetooth* escalável. Também é possível criar um servidor *Bluetooth* em dispositivo móvel, que recebe conexões de clientes que transmitem por TCP ou UDP. A essa estrutura demos o nome de serviço de inter-operação de redes.

5.4 Outros Serviços

Alem dos serviços já apresentados, o *middleware* **GATE** conta com outros serviços complementares e inerentes a sua estrutura. Esses serviços especializados são:

- **Criação de ambientes**: fornecer meios para o desenvolvimento de ambientes virtuais, disponibilizando mecanismos de controle, visualização e persistência de dados;
- **Escalabilidade de Clientes**: permitir o crescimento escalável do número de clientes que acessam o servidor;
- **Análise de tráfego**: analisar o tráfego de informações entre os clientes, permitindo realizar inferências sobre a comunicação entre os mesmos;

5.4.1 Criação de ambientes

O objetivo central do **GATE** é oferecer serviços para ambientes virtuais multiusuários. Um serviço primordial que esse tipo de aplicação necessita é a existência de um ferramental que facilite a criação desses ambientes. Uma ferramenta como deve prover meios para controlar as mudanças do ambiente, a persistência dos seus dados e visualização dos mesmos. Essas características nos leva a propor o serviço de construção de ambientes. Esse serviço funciona baseado na arquitetura da figura 5.7.

A arquitetura geral do PercepCom é mostrada na figura 5.7. Esta arquitetura segue o paradigma MVC [Buschmann et al. 1996] e o componente Armazenamento de Dados, Visualização Gráfica e Controle, representam respectivamente os módulos *Model*, *View* e *Controller* do MVC .

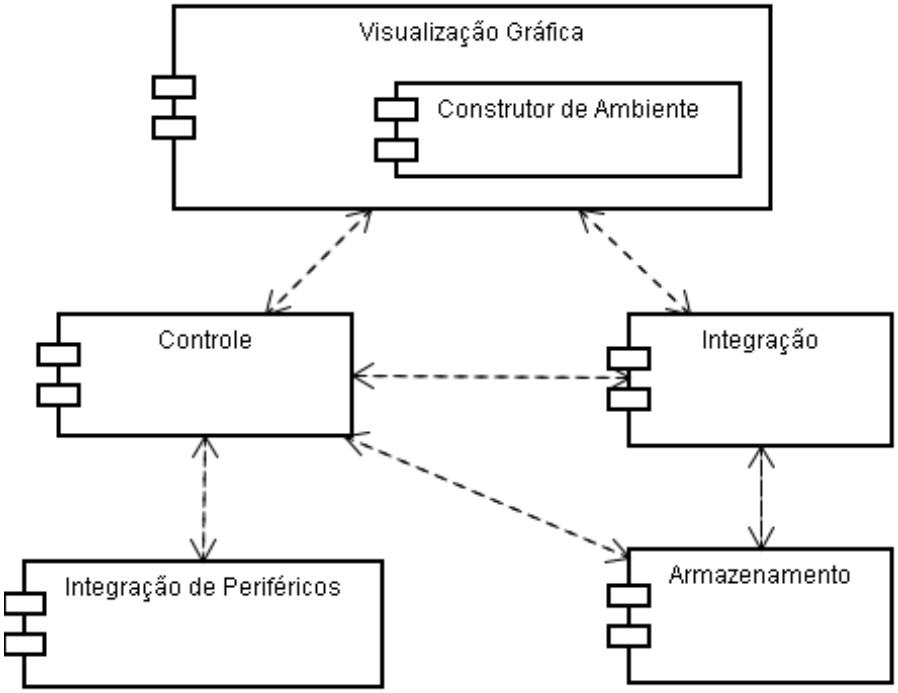

Figura 5.7: Arquitetura do serviço de criação de ambientes.

O módulo de Integração de Periféricos representa uma parte do controle que foi transformada em um módulo menor para diminuir a complexidade desta parte do sistema. Do mesmo modo, o módulo de Construção de Ambientes é uma parte especializada do módulo de Visualização, dedicada à tarefa de construção do ambiente virtual. O módulo de Integração tem a função de encapsular o sistema e permitir ao usuário abstrair o funcionamento de baixo nível do sistema. A presença do módulo Integração é uma analogia ao padrão de projeto *Facade* (Faixada) [Gamma et al. 2000].

O uso desse serviço permite construir ambientes com interfaces 2D, 3D e 1D. Esse serviço funciona junto com serviço de adequação de percepção, permitindo a integração dos ambientes com dimensão geométrica distinta. Além disso, os ambientes criados são massivos graças ao uso do **H-N2N**, que fornece o serviço de escalabilidade.

5.4.2 Escalabilidade

No projeto do nosso *middleware* utilizamos o arcabouço **H-N2N**. Esse arcabouço foi reformulado para prover o serviço de inter-operação de clientes e redes, dando origem ao **H-N2NInt**. Ambas as versões do arcabouço são capazes de prover uma arquitetura hierárquica de servidores, possibilitando a criação de ambientes virtuais massivos. A Tabela 5.3 apresenta um conjunto de mensagens do **H-N2N** que visam proporcionar o crescimento em escala do ambiente.

Quando o servidor completou seu número máximo de conexões ele envia aos novos clientes que tentam se conectar, uma mensagem *MBusy*. O servidor em resposta ao envio do *MBusy* dispara também uma mensagem *MSon*, procurando por clientes capazes de receber um cópia do servidor e atuar como tal. O cliente ainda pode informar sobre qualidade do serviço num dado instante com *MQoS*. O servidor pode ao receber essa mensagem enviar uma *MBusy*, mesmo que ainda seja capaz de receber novas conexões. Isso pode ocorrer caso a conexão já esteja comprometida com a quantidade atual de clientes.

ID	**Categoria**	**Nome**	**Descrição**
H-N2N5	Controle	MBusy	Servidor informa que não pode receber novas conexões
H-N2N6	Controle	MPing	Mensagem de Ping aos clientes
H-N2N7	Controle	MPong	Mensagem de Pong em reposta ao Ping
H-N2N12	Controle	MSon	Busca por clientes capazes de atua como Servidor
H-N2N13	Controle	MQoS	Clientes informa a QoS

Tabela 5.3: Mensagens complementares do arcabouço H-N2N

A partir da inserção de um componente de *software* na estrutura do arcabouço **H-N2N**, implementamos a idéia de uma aplicação de interpercepção. Por usar esse arcabouço, essas aplicações já eram escaláveis. Agora que a interpercepção foi incorporada ao **GATE** como um serviço, é preciso incorporar também o **H-N2N**, já que a interpercepção é parte dele. Essa incorporação faz surgir o serviço de escalabilidade de clientes. Esse serviço permite aumentar gradativamente o número de clientes conectados ao sistema.

5.4.3 Análise de Tráfego

As mensagens *MPing* e *MPong* apresentadas na Tabela 5.3 são para saber o quanto demora uma mensagem para ser transmitida pelo cliente. Seu intuito é gerar estatísticas de rede. Com base nessas estatísticas criamos o serviço de análise de tráfego. Através desse serviço é possível verificar a conexão dos usuários e fazer mudanças ou sugestões de conexão para os clientes.

Foi desenvolvida para complementação do *framework*, uma ferramenta de supervisão de redes de modo a colher dados do funcionamento da rede e sua posterior exibição para um supervisor humano de rede. A ferramenta se baseia em uma chamada ao *Aplication-Server* do **H-N2N**. Nessa chamada é colhida a quantidade de bytes enviados e recebidos por cada SlaveServer. Também é medido tempo gasto por cada servidor para o envio de um pacote simples para seus respectivos clientes (RTT, que é o acrônimo de *Round-Trip Time*).

Os dados colhidos pela ferramenta são armazenados em arquivo. Esse arquivo é usado como entrada para uma aplicação gráfica. Os dados do arquivo são posteriormente exibidos pela interface dessa aplicação. Na interface dessa aplicação, chamada de analisador, é possível verificar quantidade de bytes enviados, de bytes recebidos e o RTT. Essa quantidade pode ser mensurada para cada estação, ou para toda a rede.

Na figura 5.8 é apresentada a interface do analisador de tráfego. Ela permite que sejam selecionados os parâmetros para a construção dos gráficos baseados no dados colhidos pela rede. Podem ser selecionados dados referentes ao volume de dados recebidos ou RTT de cada cliente. Há a possibilidade também de se exibir o volume de dados de toda a rede e bem como RTT médio desta.

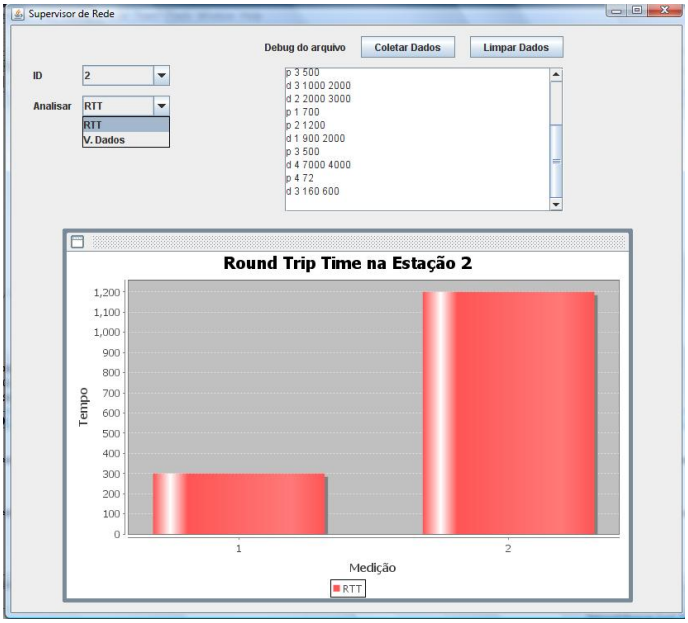

Figura 5.8: Interface do analisador de tráfego

5.5 Considerações Finais

A reunião dos serviços apresentados nesse capitulo forma o *middleware* **GATE**. Sua estrutura final é apresentada na Figura 5.9. Nessa figura os serviços do **GATE** são organizados em camadas. Os serviços ainda estão grupados conforme o tipo de serviço. Os serviços de inter-operação foram incluídos no mesmo grupo do serviço de escalabi-

lidade. Eles formam o grupo dos serviços de arcabouço. Já o serviço de conversão de dimensões foi agrupado ao serviço de construção de ambientes formando a classe dos serviços de percepção visual. Os serviços de acessibilidade foram agrupados da forma como descritos nesse capitulo.

Figura 5.9: Serviços do *Middleware* GATE organizados em camadas.

O serviço de escalabilidade está na base do *middleware* por está ligado a infra-estrutura básica onde os demais serviços são acoplados. O serviço de análise está ligado ao funcionamento do arcabouço e por isso está na camada logo acima. Nessa mesma camada está o serviço de primordial de construção de ambientes. Como o mesmo depende da infra-estrutura do arcabouço está alocado na camada acima dele.

A terceira camada é formada pelos serviços que interligam os usuários. Essa camada resolve os problemas de diversidade dos ambientes heterogêneos. Na quarta camada estão os serviços que aprimoram a interação dos usuários, conferindo maior acessibilidade ao ambiente.

Os serviços descritos nesse capitulo foram colocados em prática, testados e comparados com outras abordagens. Os resultados desses testes são o tema do próximo capítulo.

Capítulo 6

Experimentos e resultados

Para validar o *middleware* apresentado neste trabalho foram implementados dois estudos de caso. No primeiro estudo de caso o **GATE** foi aplicado na construção de uma aplicação chamado de Percepcom1D multi-clientes. No segundo estudo de caso o **GATE** foi aplicado na construção de uma ferramenta de criação de ambientes virtuais colaborativos na forma de museus, chamado GTMV.

Além dessas duas aplicações, realizamos um estudo comparativo entre o antigo arcabouço **HN2N** e novo **HN2NInt**. Esse estudo enfatiza as melhorias alcançadas com o novo arcabouço. A comparação dos arcabouços inicia a demonstração de resultados desse capítulo.

6.1 Comparações entre arcabouços

O arcabouço **HN2N** permite a criação de aplicações multiusuário massivas. Essas aplicações, contudo era dependente da plataforma Java, plataforma usada para criação do arcabouço. O **HN2N** foi reformulado para permitir a criação de clientes em outras linguagens. O objetivo dessa nova propriedade é permitir a criação de aplacações clientes capazes de executar em dispositivos dependentes de plataformas que não fossem Java. Dessa reformulação nasceu o novo arcabouço **HN2NInt**. Além dessa melhoria fundamental, ele possui outras melhorias secundárias listadas na Tabela 6.1.

Na Tabela 6.1 é traçado um comparativo entre o arcabouço **HN2N** e a sua reformulação **HN2NInt**. As características comparadas nessa tabela dizem respeito ao servidor. Ambos os arcabouços tem a propriedade de facilitar o processo de criação de servidores para aplicações distribuídas. Como evidencia a Tabela 6.1, o arcabouço reformulado é melhor em todas as características.

O arcabouço reformulado permite a criação de servidores mais simples e intuitivos. Isso ocorre, pois seus servidores possuem menor dependência de outras classes. Além disso, no processo de criação as mensagens recebem uma identificação, o que as torna passiveis de classificação.

Para garantir a inter-operação, o arcabouço reformulado cria aplicações que lêem e escrevem bytes. Esse processo de leitura e escrita é conseguido através de classes auxiliares (já discutidas no capítulo anterior). Analisando o diagrama de classes do arcabouço reformulado (apresentado no capítulo anterior) é possível constatar que essas novas clas-

Característica	**HN2N**	**HN2NInt**
Dependência	Necessário estender diversas classes para se construir um servidor	Constrói o servidor e registra os listeners necessários
Leitura e Escrita	Lê e escreve objetos	Lê e escreve bytes
Criação de mensagens	Mensagens são criadas estendendo a classe Message	Mensagens são lidas e escritas utilizando classes auxiliares
Classificação das mensagens	Definições dos tipos de mensagens internas e de aplicações podem dar conflitos	Definições dos tipos de mensagens internas e de aplicações são independentes entre si
Identificação das mensagens	Mensagens com o mesmo identificador	Diferencia mensagens internas de mensagens da aplicação
Tamanho da Mensagem	241 bytes	12 Bytes

Tabela 6.1: Comparação entre os arcabouços, sobre o aspecto do servidor.

ses auxiliares chegam a tornar o processo de leitura e escrita mais complexo. Contudo, a perda em simplicidade estrutural é recompensada em versatilidade, já que o novo arcabouço permite novos serviços (como a inter-operação).

O novo modelo de criação das mensagens, trás o ganho mais significativo: diminuição no tamanho das mensagens. As mensagens do arcabouço reformulado se tornaram cerca de 20 vezes menores. Isso permite que as aplicações criadas com o arcabouço reformulado sejam executadas em ambientes com limitação de espaço em disco (como um sistema embarcado, por exemplo).

Na Tabela 6.2 é feito outro estudo comparativo mostrando as mudanças no aspecto do cliente. A comparação novamente evidencia a sensível melhoria entre o arcabouço modificado e o arcabouço original.

Como pode ser visto na Tabela 6.2, uma melhora importante está na possibilidade criar aplicações cliente em diferentes linguagens, motivo que levou a reformulação do arcabouço. Esse mesmo aspecto permite a criação de aplicação para dispositivos com poucos e recusros e com limitações de linguagem.

Outro ponto de destaque está na sensibilidade do cliente às mudanças do ambiente. Antes a sua baixa sensibilidade tornava o cliente lento pois o mesmo deveria constantemente perguntar se estava ou não conectado. Além disso, o cliente antes não tinha conhecimento quando a conexão havia caído, e agora o mesmo é avisado através de um evento, gerado por uma mensagem interna de controle. Essa mesma sensibilidade do arcabouço reformulado, permite ao cliente ter a noção de quando houve uma mudança de grupo. Em síntese, os clientes criados com o novo arcabouço são mais sensíveis às mudanças do ambiente e ganham em desempenho se comparados aos antigos clientes.

Características	HN2N	HN2NInt
Linguagem	Apenas aplicações escritas em Java	Aplicações escritas em linguagens com suporte a dados binários (C, C++, C#, etc.)
Dispositivo	Aplicações para Desktop	Aplicações para Desktop, Dispositivos móveis e dispositivos embarcados (Set-top Box)
Construção	O cliente é construído como objeto	O cliente é construído através de um método estático
Tratamento de estado	Tem que ficar perguntando se está conectado	É necessário um listener para ouvir eventos internos que representam estados
Queda de conexão	Aplicação não sabe quando a conexão foi quebrada	Eventos são gerados indicando o estado da conexão
Mudança de grupo	Aplicação não sabe quando mudou de grupo	Eventos são gerados indicando o estado da mudança

Tabela 6.2: Comparação entre os arcabouços sobre o aspecto do cliente.

6.2 O Percepcom1D multi-clientes

O Percepcom1D é uma aplicação textual para simulação de ambientes virtuais. Essa aplicação foi construída com os serviços de inter-operação de clientes, construção de ambientes e análise de tráfego. Visando validar esses serviços foi realizado um experimento, onde 6 clientes acessam o ambiente num determinado e interagem.

Nessa aplicação, estão disponíveis aos usuários as seguintes ações: conversar, listar usuário, observar. Conversar envia uma mensagem com o texto "Oi" para os demais usuários. Listar retorna a lista dos usuários presentes no instante que a ação foi invocada. Observar retorna uma descrição sobre a aparência do usuário. Essa descrição é fornecida pelo próprio usuário. Como passo inicial da validação, implementamos um servidor em linguagem Java e um conjunto de aplicações cliente: um cliente utilizando a linguagem J2ME (na verdade uma API Java para aplicativos de celular) e um cliente applet utilizando a linguagem Java.

No experimento, o servidor é executado em um PC e os vários clientes se conectam ao mesmo. O cliente *Applet* é executado através de qualquer navegador da *Internet*. A aplicação para celular pode foi executada através de simulador presente no kit de desenvolvimento do J2ME. Figura 6.1 mostra a interface desses clientes.

Durante o teste a ferramenta de análise gera um arquivo de log automaticamente. Esse arquivo serve de entrada para geração de gráficos sobre o tráfego. È possível gerar gráficos sobre o RTT e sobre o volume de dados trocado. Ambos os gráficos podem ser desenhados para toda a rede (todos os clientes) ou para cada estação em separado. Na Figura 6.2 mostramos o gráfico que mede o RTT de toda a rede, formada nesse experimento por 6

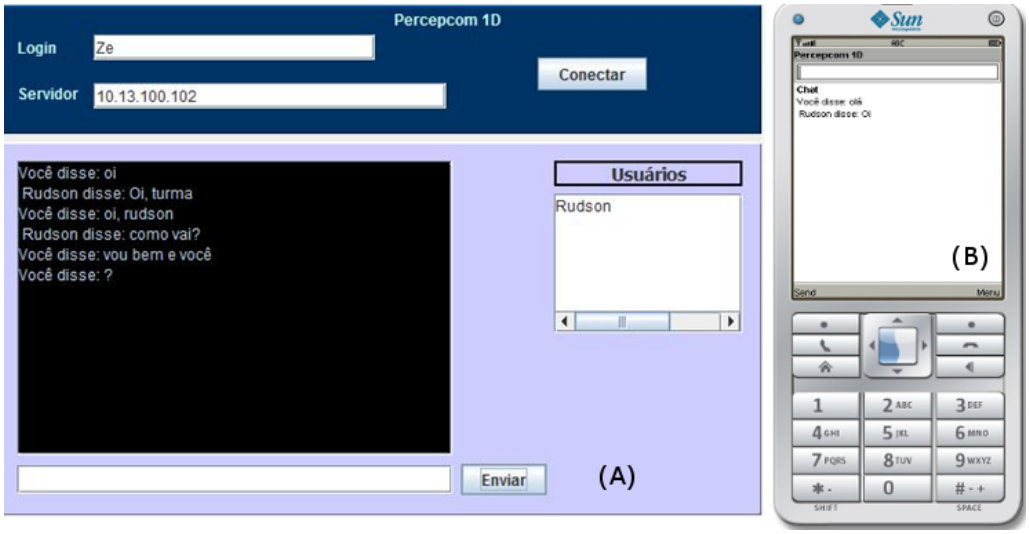

Figura 6.1: Interface dos clientes da aplicação multi-clientes. (A) aplicação applet. (B) aplicação celular.

clientes.

No experimento analisado pela Figura 6.2, 3 clientes são applets (estações com números pares, incluindo o 0) e 3 são uma aplicação J2ME (estações impares). Podemos ver que os clientes do tipo *applet* possuem um RTT consideravelmente menor em comparação com os clientes J2ME. A estação 2, por exemplo, teve um RTT médio de 2 milissegundos, contra 15 milissegundos da estação 3, que executa um cliente J2ME. Devido à grande diferença entre os tempos das aplicações com applet em comparação com as aplicações para celular, isso nos leva a sugerir uma mudança nos clientes do tipo J2ME. Contudo, neste caso nosso intuito é apenas mostrar a capacidade do GATE em oferecer o serviço, e, portanto a inferência desses resultados caberia a uma análise futura por parte dos usuários desse serviço.

6.3 O projeto GT-MV

O GT-MV (Grupo de Trabalho de Museus Virtuais) é um projeto que visa oferecer aos usuários da RNP um conjunto de ferramentas para construção e manutenção de museus virtuais. Um dos principais objetivos deste projeto é garantir que estas ferramentas sejam de fácil utilização. Além disso, as ferramentas propostas são executadas diretamente de navegadores da *Internet* e estão integradas com o banco de dados do sistema.

Através deste serviço, todas as instituições interessadas na construção de versões virtuais de museus podem solicitar o acesso ao conjunto de ferramentas do GT-MV. É disponibilizado a um usuário, chamado aqui de curador do museu virtual, uma conta e uma aplicação cliente (versão do curador) que permite ao mesmo construir o museu e submetê-lo ao fornecedor do serviço. O fornecedor do serviço deve manter o mesmo em um servidor dedicado a esta tarefa. Um endereço ficará disponível em um portal onde este museu poderá ser acessado posteriormente por qualquer visitante.

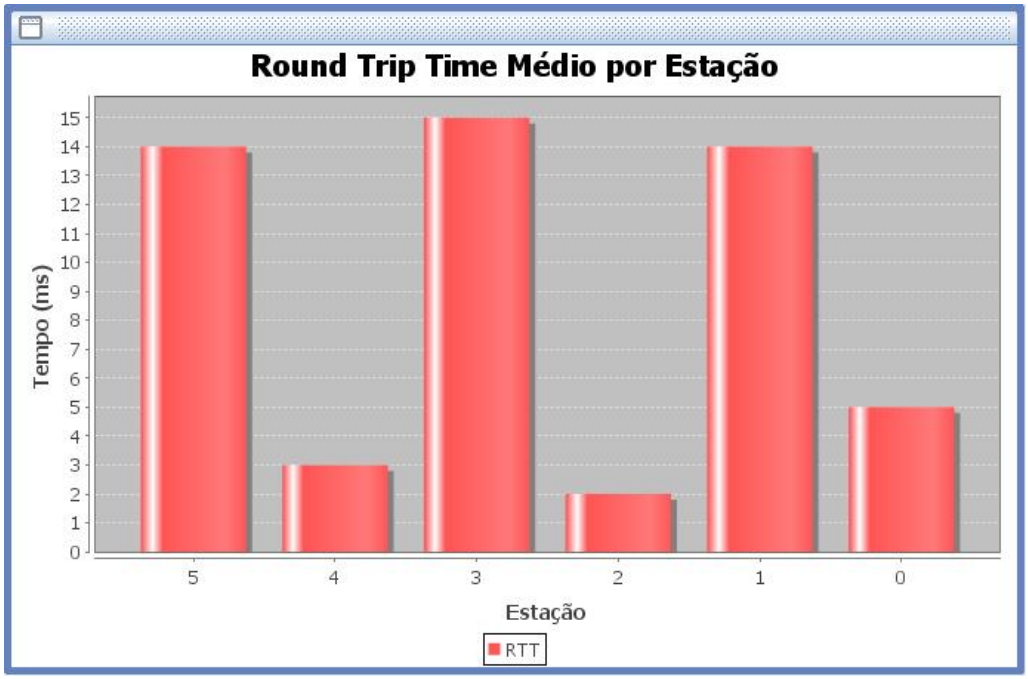

Figura 6.2: Analise do RTT de todos os clientes do experimento Percepcom1D

Os visitantes de um museu virtual construído com as ferramentas do GT-MV navegam pelo mesmo utilizando um avatar animado. Essa representação virtual do visitante permite que ele seja visualizado pelas demais pessoas que estão visitando o museu ao mesmo tempo. Todos os visitantes desses museus também podem conversar através de uma ferramenta de chat (bate-papo) que está acoplada à interface gráfica da versão do visitante.

O ambiente multiusuário disponibilizado aos visitantes dos museus construídos pelo aplicativo do GT-MV é um dos grandes diferenciais entre esse aplicativo e outros aplicativos de museus virtuais. Os demais aplicativos não permitem essa interação entre os visitantes. Outro diferencial do aplicativo GT-MV é a sua facilidade de uso que não requer do curador conhecimento técnico para utilizar as ferramentas disponíveis. Integrado ao serviço do GT-MV, existe também um conjunto de módulos de *software* que permitem a integração das ferramentas do mesmo com dispositivos robóticos.

O GT-MV foi construído com base no midddleware **GATE**. Ele utiliza os serviços de Escalabilidade, Construção de Ambientes, Inter-operação de Clientes e Conversão de dimensões. Além disso, foi estabelecido dois novos serviços: comunicação com sensores e criação de guias autônomos.

6.3.1 Comunicação com sensores

O serviço de comunicação com sensores permite a integração de acelerômetros, câmeras e outros tipos de sensores ao sistema. Esses sensores são usados para comunicação com os dispositivos robóticos. A arquitetura desse serviço é mostrada na Figura 6.3.

Os sensores (apresentados como nós no diagrama da Figura 6.3) são dispositivos que

alimentam o sistema com dados. Esses dados são processados no módulo de Processamento. Nesse módulo, o dado é recebido pelo Receptor de Comandos. Os sensores enviam sinais que são mapeados em ações. Essas ações se tornam comandos que são repassados ao cliente que utiliza o serviço.

Sensores do tipo Câmera enviam imagens para o módulo de Processamento. Essas imagens são processadas pelo Emissor de Imagens. As imagens formam um fluxo de vídeo. O Emissor organiza o vídeo em num buffer que depois é enviado para o Servidor do Sistema.

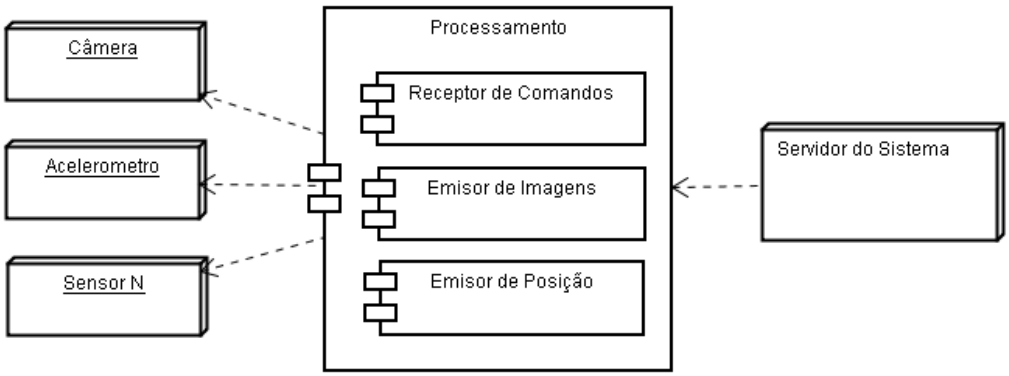

Figura 6.3: Arquitetura do Serviço de Comunicação com Sensores

Os Sensores do tipo acelerômetro enviam vetores de coordenadas. Esses vetores geralmente tridimensionais. Essa posição segue o modelo do **GATE** e é representado por vetor homogêneo 3D. No módulo de processamento, esse vetor é convertido numa posição 3D, pelo Emissor de Posição. Esse emissor gera pontos para a movimentação de avatares de um dispositivo robótico. Isso permite movimento de um robô seja representado no ambiente virtual. Assim é possível criar uma aplicação de realidade mista, onde o real e o virtual interagem simultaneamente.

6.3.2 Serviço de Criação de Agentes

Esse serviço possibilita a criação de um agente com IA (Inteligência Artificial) que age de forma autônoma. Esse agente é representado com um avatar. Esse avatar trafega pelo ambiente seguindo um roteiro definido na sua criação. Esse roteiro é baseado num caminho (*path*) e num tipo de comportamento. O caminho é formado por um conjunto de pontos. O comportamento envolve ações a diálogos pré-definidos. A Figura 6.4 mostra a arquitetura desse serviço.

O Criador de caminhos é usado para definir locais dentro do ambiente por onde o agente irá trafegar. Esses locais representam posições dentro do ambiente virtual. A união de todos esses pontos forma um caminho que o agente irá percorrer. O Criador gera ao final de sua execução um vetor de posições. Esse vetor é usado depois pelo Animador para criar a animação de movimento do avatar.

Além dos caminhos que o agente percorre, sua atividade dentro ambiente pode ser moldada por um comportamento. O Seletor de Comportamento oferece a possibilidade

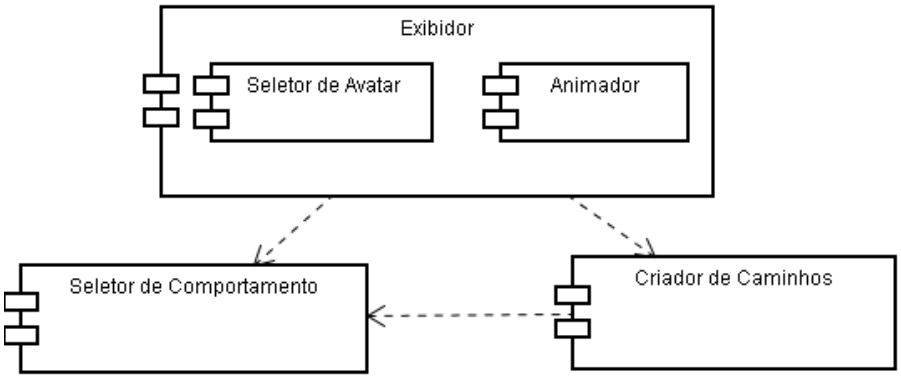

Figura 6.4: Arquitetura do serviço de criação de agentes

de definir o que o agente fala, através de arquivos de texto. Arquivos de áudio também podem ser associados ao dialogo do agente. Permite também alterar a velocidade de sua fala e do seu caminho. Ainda é possível inserir ações como parar numa certa posição e esperar por alguma interação do usuário.

6.3.3 Arquitetura do GTMV

A Figura 6.5 mostra um diagrama de componentes com a arquitetura do GT-MV. Nessa arquitetura vemos 2 servidores (Multi-usuário e Controle Remoto). Esses servidores foram construídos com os serviços de Escalabilidade e Construção de Ambientes. Os 5 clientes (Controle Remoto *Web*, Controle Remoto Mobile, Visitante, Curador, 3D Robô) foram construídos com os serviços de inter-operação de clientes. O *Streamer* de Vídeo é uma representação direta do componente Emissor de Imagens, presente no serviço Comunicação com Sensores.

Os clientes com o nome Controle, ainda utilizam o serviço de Comunicação com Sensores. Os clientes Curador e Visitante utilizam também o serviço de construção de ambientes.

O GT-MV ainda conta com outros componentes, que não estão diretamente ligados aos serviços do **GATE**. O Sistema Web é composto por um aplicativo e um servidor. Ambos executam acima das camadas do **GATE**. Apenas esse sistema tem acesso ao Banco de Dados. Por isso ele atua como interface entre os Clientes e o Banco de Dados.

O sistema ainda conta com uma aplicação chamada Guia Virtual. Esse aplicativo é construído com o serviço de Criação de Agentes Autônomos. Ele permite a criação de agentes autônomos que atuam como guias do museu. Esses guias conduzem os visitantes pelas obras percorrendo um trajeto definido através do Cliente Curador. O resultado pode ser exibido tanto para o Cliente Curador, quanto para Cliente Visitante.

O Cliente Curador possui 3 ferramentas para autoria de museus virtuais. Uma ferramenta permite a criação da estrutura física do museu. No outro é possível editar o museu, inserindo obras, moveis, modificando a cor e a textura das paredes. As obras podem ser quadros, vídeos e músicas. Na terceira ferramenta é possível criar guias virtuais com base no serviço de criação de agentes autônomos.

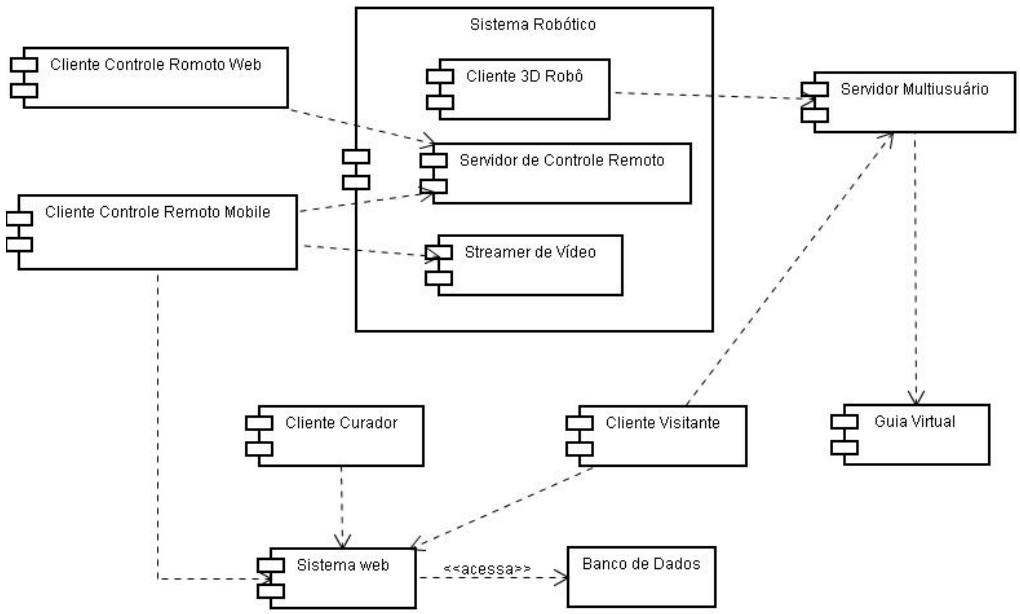

Figura 6.5: Arquitetura do GT-MV

O serviço de conversão de dimensões está inserido no Cliente Curador. A ferramenta de criação de museus usa interface 2D. Ele representa a planta baixa do museu. O serviço de conversão de dimensões transforma o mapa 2D desenhado na ferramenta de construção em um mapa 3D. Esse mapa 3D é mostrado na tela do editor de museus. A visão dessa tela é a mesma do Cliente Visitante. A Figura 6.6 mostra a interface da aplicação GTMV: na Figura (A) temos o construtor e na (B) o editor.

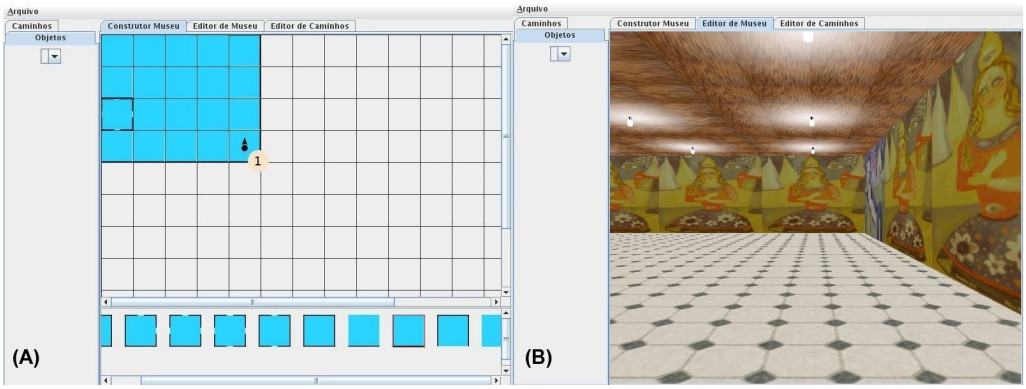

Figura 6.6: Interface da aplicação GTMV. (A) Interface do Construtor de museus. (B) Interface do Editor de museus.

Na Figura 6.6 (A) é visto o construtor divido em dois painéis. O painel superior do construtor é desenhado em forma de grade. Nessa grade o usuário pode adicionar blocos. Esses blocos representam modelos de salas. Esse modelos são diferenciados pela quantidade de paredes, portas e outras estruturas de uma sala. Temos por exemplo, modelos com 4 paredes e uma porta, 4 paredes e 4 portas, 2 paredes sem portas, entre

outras possíveis combinações. Os blocos estão reunidos no painel inferior, formando uma espécie de paleta de blocos. Os usuários arrastam os blocos até a grade do painel superior. A grade do painel principal é representada por uma matriz. Os modelos de sala 2D que são associados a uma modelagem tridimensional. Uma tabela hash faz o mapeamento entre a peça e sua modelagem 3D. Ao adicionar um bloco 2D a grade um evento é gerado. O evento captura os índices (i, j) da posição na matriz da grade aonde a peça foi adicionada. A posição tridimensional da peça é então calculada a partir dos índices i, j da matriz que representa a grade. Com esses índices, calculamos a posição tridimensional com equação de transformação 6.1.

$$(x, y, z) = (i * Ls - Ls/2, 0, j * Ps - Ps/2) \tag{6.1}$$

Na equação 6.1 a constante Ls representa a largura da sala e Ps a profundidade da mesma sala. Subtrai-se a metade da largura da sala e metade da profundidade da sala para ajustar os valores com o centro da sala. Na parte bidimensional, os índices indicam a posição do lado superior esquerdo da sala.

No mesmo evento capturado no construtor temos o bloco envolvido e a ação que está sendo feita. A ação pode ser adição, remoção ou rotação. Com esse bloco, buscamos a sua modelagem 3D. Tendo o mesmo é possível então fazer as operações necessárias, agora no modelo tridimensional. O resultado sobre o modelo 3D é adicionado a visualização do editor de museus.

Na Figura 6.6 (B) vemos a representação 3D do museu construído na Figura 6.6(A). O círculo com número 1 na Figura 6.6 (A) mostra um símbolo usado para marcar o ponto inicial do museu. É neste ponto que a visão do 3D é colocada e com a mesma orientação prevista no construtor.

6.3.4 Aplicação de Realidade Mista

Os novos serviços criados para o GTMV foram utilizados em um experimento de dança telematica. Durante o espetáculo e-pormundos Afeto, os museus virtuais foram usados em tempo real como pano de fundo do espetáculo. Através de projeções e jogo de luzes foi possivel criar o efeito de que os dançarinos e avatares estavam no mesmo espaço. Uma vez que o ambiente era projetado em tempo real, os visitante participaram do próprio espetáculo com sua navegação pelo ambeinte. Alem disso, os visitante puderam interagir com os demais, através do *chat*. Isso tudo proporcionou uma experiência única de realidade mista. Cerca de 100 visitantes remotamente espalhdos assitiram e interagiram com o espetáculo, durante suas duas apresentações. A Figura 6.7 mostra uma cena do espetáculo.

Além da participação virtual de pessoas, o espetáculo contou também com a participação de um robô. Atuando ao lado de bailarinos, o robô Galatéia (modelo Pionner3 da Active Robotics) se desclocava em cena a partir de entradas de um sensor acoplado no corpo de uma da bailarinas. O sensor usado era uma controle do *Nintendo Wii*. No espetáculo o robô estava na cidade de Natal, enquanto a bailarina estava em Fortaleza. Um outro grupo de artistas dançava na cidade de Barcelona, na Espanha. Todos, visitantes virtuais, daçarinos e robô, participaram juntos de um grande ambiente de realidade mista,

Figura 6.7: Espetáculo e-pormundos

provido pelos serviços do GATE.

6.3.5 Avaliação do GTMV

A aplicação do GTMV permite a criação de ambientes virtuais colaborativos. Ela foi criada com serviços oferecidos pelo *middleware* **GATE**. A fim de validar o **GATE** como uma abordagem útil a esse tipo de aplicação foi realizada uma avaliação funcional da aplicação GTMV. Ao mesmo tempo em que avaliamos, é traçado um comparativo entre a aplicação do GTMV com outra aplicação similar criada com *middleware* CTU (citado como trabalho relacionado no capítulo 3).

A avaliação funcional apresentada consiste na análise das funções oferecidas por ambas as aplicações em termos de completeza. O parâmetro completeza avalia se as funcionalidades oferecidas executam suas tarefas de forma a contemplar todas as expectativas para aquela funcionalidade. Essa é uma das métricas de modelo de projeto orientado a objetos definidas por Whitmire [Whitmire 1997]. A avaliação foi realizada por uma equipe específica de testes, que não atuou no desenvolvimento do GATE. Foi realizada uma avaliação com o GATE e outra com o CTU. Nas avaliações, as funcionalidade foram pontuadas de 1 a 5, onde esses valores representam:

1. Não realizado (a funcionalidade não foi implementada);
2. Realizado, mais incompleto;
3. Completo, mas com dependências externas (depende de outros módulos de *software*, além do *middleware*);
4. Completo e sem dependências externas;
5. Totalmente completo (ou seja, contempla o nível 4) e otimizado.

A Tabela 6.3 mostra os resultados da avaliação funcional das aplicações. Nessa tabela são listadas funcionalidades que deveriam está presente em ambas as aplicações. As funcionalidades consideradas nível 4 oferecem ao usuário um serviço coeso e auto contido. Em ambas as aplicações, isto é conseguido, isto é, alcançado, pelas funções Navegar, comunicação com áudio e interação com sensores.

Funcionalidade	**CTU**	**GATE**
Navegar pelo museu 3D	4	4
Representação por avatar	5	4
Comunicação com áudio	4	4
Criação da estrutura do ambiente	3	5
Inserir objetos no ambiente criado	3	5
Interação com sensores	4	4
Uso de agentes autônomos	1	4
Percentagem de completude	**68,5%**	**85,7%**

Tabela 6.3: Avaliação funcional comparativa entre o *middleware* **GATE** e o CTU.

As funcionalidades que consideramos nível 5 são acrescidas de facilidades no intuito de deixar o serviço ainda mais atrativo. No caso da função representação por avatar, o CTU permite a representação com vídeo, ao contrário do **GATE**. No caso das funções de Criação e Inserção de objetos o **GATE** permite adicionar arquivos áudio, vídeo, além de quaisquer objetos 3D. Já o CTU provê apenas a inserção de objetos 3D e tanto a função de criação, quanto a de inserção não são serviços garantidos diretamente pelo *middleware*.

O serviço de uso de agentes autônomos não chega a ser garantido pelo CTU. Esse serviço é fundamental para ambientes multiusuários, principalmente para criação de simulações e testes.

Como resultado da avaliação funcional, na última linha da Tabela 6.3 é apresentado à porcentagem de completude do sistema. Fazendo-se uma análise geral da avaliação funcional temos uma percentagem 85,7% para completude da aplicação criada com **GATE**, contra 68,5% da aplicação criada pelo CTU. Isso mostra um alto nível de aderência entre funções propostos com as funcionalidades implementadas por nossa aplicação, em detrimento da aplicação criada com o outro *middleware*.

Capítulo 7

Conclusões

Os desenvolvimentos da área de computação pervasiva são promissores, contudo, as qualidades sugeridas pelos idealizadores desta área ainda são difíceis de serem alcançadas. O *middleware* aqui proposto visa reunir algumas dessas qualidades para criar um sistema pervarviso para aplicações de realidade virtual, passível de utilização em outras aplicações envolvendo o uso da *Internet*.

Acreditamos que, com a utilização da *Interpercepção*, barreiras tecnológicas possam ser quebradas ao permitir que um usuário acesse um ambiente virtual ou mesmo um jogo a partir de uma gama de dispositivos diferentes. Além de ter liberdade para escolher um dispositivo que melhor se adapte as suas condições financeiras, este usuário ainda terá a garantia de que poderá interagir com todos os outros usuários que estão acessando o mesmo ambiente.

Por se adequar ao perfil de cada usuário, um aplicativo de software que utilize a *Interpercepção* é comercialmente viavel, pois permite o aumento da gama de usuários de qualquer aplicação. Um aplicativo que antes só era possivel num PC, agora poderá ser possivel para um usuário de PDA, de celular e mesmo da TVDI. Para a indústria do entretenimento, a *Interpercepção* entre dispositivos tambem possibilitará um novo paradigma na contrução de jogos para multiplas plataformas, pois possibilitará que as versões de cada plataforma sejam integradas para forma um mesmo ambiente de jogo *muli-player*.

O enfoque de acessibilidade dentro da Intrepercepção entre dispositivos tem uma forte aplicação social, pois favorece a inclusão digital. Usuários que antes ficariam de fora de algum tipo de aplicativo colaborativa, poderaão ter acesso ao mesmo graças a Interpercepção. Usuários com debilidades visuais tambem poderão intergir com o sistema atraves de sistese e reconhecimento de voz, o que reforça mais ainda a aplicação social da *Interpercepção* entre dispositivos. Este usuário não terá só a oportunidade de acessar o ambiente, ele também terá o direito de interagir com os demais de forma síncrona.

Ao longo do desenvolvimento desse trabalho foram elaborados vários artigos. Desses artigos artigos foram 3 foram publicados em conferências internacionais [Dantas et al. 2008, Burlamaqui et al. 2008, Azevedo et al. 2008] relevantes. Foi também elaborado e publicado em um artigo em um relevante *Journal* [Burlamaqui et al. 2009] da área. Outros artigos estão em elaboração nesse momento, com resultados ainda mais recentes do nosso trabalho.

7.1 Trabalhos futuros

Nossa idéia é que a *middleware* aqui proposta evolua para um arcabouço. Esse arcabouço deverá permitir a criação de aplicações *interperceptivas* multiúsuário que são executadas em cada uma das plataformas para qual o *middleware* foi pensado. Esse arcabouço fornece além dos serviços aqui propostos um conjunto de ferramentas de apoio a criação de aplicações interperceptivas. Essas ferramentas envolvem a criação automatizada de ambientes virtuais. Baseado na existência de um conjunto de meta-dados, essa ferramenta geraria sem intervenção humana, um ambiente em alguma dimensão geométrica. A ferramenta aqui proposta é na verdade uma evolução da ferramenta já desenvolvida para o projeto GTMV.

Os serviços do **GATE** foram usados com sucesso no desenvolvimento do projeto GTMV. Agora com o término do projeto, esse serviço deverá também ser usado em um novo projeto chamado GTRM (Grupo de Trabalho de Realidade Mista). Nesse novo projeto os serviços do GATE irão prover infra-estrutura para criação de uma aplicação de tele-aula virtual. Nessa aplicação, alunos presencias e a distância assitem a mesma aula em tempo real. Na sala, preparada com câmeras o vídeo é capturado em tempo real e transmitido pela rede para os usuários remotos. Os usuários remotos têm a opção de assitir a aula através de vídeo, ou acessando um ambiente virtual.

Pela definição dos graus de *Interpercepção*, ainda não foi possível estabelecer um serviço para alcançar o 4^{o} grau de *Interpercepção* (comunicação inter-cultural). Idealizamos para esse serviço o uso de avatares animados que usam gestos. Esses gestos são baseadas em representações de expressões populares de várias culturas. As expressões formariam um vocabulário que é traduzido pelos avatares. As ferramentas de acessibilidade por conversação ainda não foram testadas do ponto de vista de usabilidade. Para testar ambos os serviços planejamos um teste envolvendo pessoas com debilidade visual e pessoas sem debilidade, afim de comprovar a total eficácia do nosso serviço na integração desses perfis de usuários.

Os serviços de Inter-operação irão demandar novos testes. Os testes realizados com aplicações na linguame C se comunicando com aplicações em outras foram realizados. Contudo, não chegaram a apresentar resultados relevantes para esposição dos resultados. Também é preciso fazer experiemtos com aplicações na linguagem Lua para o ambiente da TV Digital. Os primeiros testes foram feitos para validar a integração, contudo, ainda não apresentam resultados significativos para redação de artigos.

Outra meta é a criação de uma IDL (Linguagem de Descrição de Interface). Essa linguagem descritiva é comum e presente nas abordagens de *middlewares*, incluindo as citadas no capitulo 4. Através de uma linguagem dessa tipo seria possível facilitar o uso do nosso *middleware* através da geração de *stubs* do cliente e servidor. Isso permitiria uma maior liberdade ao desenvolvedor para criar suas aplicações. Na atual versão, é necessário que o desenvolvedor realize parte da contrução da infra-estrutura de comunicação da sua aplicação com o serviço.

Como importante aplicação futura, pretendemos usar o GATE para criar um novo conceito de *wiki* de ambientes virtuais. Similar a uma aplicação *wiki*, essa nova aplicação permitirá a criação e edição colaborativa de ambientes virtuais através da *Internet*. Além

disso, essa *wiki* contará com serviços que contemplam todos os graus da *interpercepção*. Com as ferramentas de criação automatizadas de ambientes, será possivel ainda constuir colaborativamente o ambiente. Assim, ao ponto que um usuário cria uma porção textual do ambiente, outro contrbui com um arquivo de áudio com esse texto, outro com a verão 2D e outro com a versão 3D. Com as várias contribuções possíveis será preciso rever a forma como os dados persitirão no sistema, com a finalidade de garantir um serviço de qualidade a todos os usuários da *wiki* de ambientes virtuais.

?

Referências Bibliográficas

Al-Jaroodi, Jameela, Ahlam Kharhash, Amal AlDhahiri, Asma Shamisi, Aysha Dhaheri, Fatima AlQayedi & Sheikha Dhaheri (2008), Collaborative resource discovery in pervasive computing environments, *em* 'International Symposium on Collaborative Technologies and Systems', pp. 135–141.

Azevedo, Samuel, Aquiles Burlamaqui, Rummenigge Dantas, Claudio Schneider, Rafael Gomes, Julio Melo, Josivan Xavier & Luiz M. G. Gonçalves (2006), Interperception on shared virtual environments, *em* 'Proceedings of IEEE VECIMS'06', IEEE, Inc., La Coruña, Spain, pp. 109–113.

Azevedo, Samuel, Julio Cesar Melo, Rummenigge Dantas, Aquiles Burlamaqui, Claudio Schneider, Josivan Xavier & Luiz Marcos Garcia Gonçalves (2008), Issues on interperceptive games, *em* 'IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems', Istambul, Turkey.

Bartle, Richard (1990), 'Early MUD history', Artigo postado para o grupo USENET rec.games.mud.

Benford, Steve, Carsten Magerkurth & Peter Ljungstrand (2005), 'Bridging the physical and digital in pervasive gaming', *Communications Of The ACM* **48**(3), 54–57.

Benford, Steve, Chris Greenhalgh, Mike Craven, Graham Walker, Tim Regan, Jason Morphett & Johh Wyver (2000), 'Inhabited television: Broadcasting interaction from within collaborative virtual environments', *ACM Transactions on Computer-Human Interaction* **7**(4), 510–547.

Bernstein, Philip A. (1996), 'Middleware: A model for distributed system services', *Communication ACM* **39**(2), 86–98.

Billinghurst, M., H. Kato, K. Kiyokawa, D. Belcher & I. Poupyrev (2002), 'Experiments with face to face collaborative ar interfaces', *Virtual Reality Journal* **4**(2).

Birrell, Andrew D. & Bruce Jay Nelson (1984), 'Implementing remote procedure calls', *ACM Transactions on Computer Systems* **2**(1), 39–59.

Burdea, Grigore C. & Philippe Coiffet (2003), *Virtual Reality Technology*, 2ª edição, Wiley-IEEE Press.

Burlamaqui, Aquiles, Guido Lemos, Jauvane Oliveira, Marlos Oliveira & Luiz M. G. Gonçalves (2006), H-N2N - uma solução escalável para ambientes virtuais colaborativos massivos, *em* 'Proceedings of XII Simpósio Brasileiro de Sistemas Multimídia e Web', Sociedade Brasileira de Computação, Natal, RN.

Burlamaqui, Aquiles, Rummenigge Dantas, Claudio Schneider, Josivan Xavier, Samauel Azevedo, Julio Melo, Rafael Gomes & Luiz M. G. Gonçalves (2006), Desenvolvimento de ambientes multi-usuários interdimensionais, *em* 'Proceedings of VII Symposium of Virtual Reality'06', Sociedade Brasileira de Computação, Bélem, PA, pp. 109–113.

Burlamaqui, Aquiles, Samuel Azevedo, Julio Cesar Melo, Rodrigo Moreira Araujo, Rodrigo Pinheiro, Antonio Cosme, Rummenigge Dantas, Claudio Schneider, Josivan Xavier & Luiz Marcos Garcia Gonçalves (2008), Indirect group interaction in interpeceptives virtual environments, *em* 'Proceedings of IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems', Istambul, Turkey.

Burlamaqui, Aquiles, Samuel O. Azevedo, Rummenigge R. Dantas, Claudio A. Schneider, Josivan S. Xavier, Julio C. P. Melo, Luiz M. G. Gonçalves, Guido L. S. Filho & Jauvane C. de Oliveira (2009), 'The H-N2N framework: towards providing interperception in massive applications', *Multimedia Tools and Applications* **45**(1-3).

Buschmann, Frank, Regine Meunier, Hans Rohnert, Peter Sommerlad & Michael Stal (1996), *Pattern-oriented software architecture: a system of patterns*, 1ª edição, John Wiley and Sons.

Curry, Edward (2004), *Message-Oriented Middleware*, John Wiley and Sons.

da Costa, Cristiano André, Adenauer C. Yamin & Cláudio Fernando Resin Geyer (2008), 'Toward a general software infrastructure for ubiquitous computing', *IEEE Pervasive Computing* **7**(1), 64–73.

Damer, Bruce (1997), *Avatars: Exploring and Building Virtual Worlds on the Internet*, 1ª edição, Peachpit Press.

Dantas, Rummenigge, Aquiles Burlamaqui & Luiz Marcos Garcia Gonçalves (2006), Interdimensionality, *em* 'Workshop de Teses de Disertações cientifica do XIX SIBGRAPI', Manaus.

Dantas, Rummenigge, Aquiles M. F. Burlamaqui, Samuel Azevedo, Julio Melo, Claudio Schneider, Josivan Xavier & Luiz M. G. Gonçalves (2008), Virtual multiverse: Interperception in multiple virtual universes, *em* 'IEEE International Conference on Virtual Environments, Human-Computer Interfaces, and Measurement Systems', Hong Kong, China.

Dantas, Rummenigge R., Cláudio Schneider, Josivan Xavier, Aquiles Burlamaqui & Luiz Marcos G. Gonçalves (2005), Um componente gráfico para o desenvolvimento de

ambientes virtuais massivos com java3d, *em* 'XI Simpósio Brasileiro de Sistemas Multimídia e Web'.

Dieterle, Edward & Jody Clarke (2007), Multi-user virtual environments for teaching and learning, *em* 'Encyclopedia of multimedia technology and networking'.

Dix, Alan, Janet E. Finlay & Gregory D. Abowd (2004), *Human-Computer Interaction*, 3ª edição, Pearson.

Dubois, Emmanuel, Philip Gray & Laurence Nigay (2009), *The Engineering of Mixed Reality Systemsn*, 1ª edição, Springer.

Ellis, Steve R. (1994), 'What are virtual environments?', *IEEE Computer Graphics e Applications* **14**(1), 17–22.

Elllis, Steve (1991), Nature and origin of virtual environments: a bibliographical essay, *em* 'Computer Systems in Engineering', pp. 321–346.

Ensor, Jim (2003), *Future Net: The Essential Guide to Internet and Technology Megatrends*, Trafford Publishing, Victoria, BC.

Erlandson, Robert F. (2007), *Universal And Accessible Design For Products, Services, And Processes*, 1ª edição, Taylor and Francis.

Fernandes, Jorge, Guido Lemos & Gleidosn Elias (2004), Introdução à televisão digital interativa: Arquitetura, protocolos, padrões e práticas, *em* 'Proceedings of Jornada de Atualização em Informática do Congresso da Sociedade Brasileira de Computação'04', Sociedade Brasileira de Computação, Salvador, BA, pp. 109–113.

Fine, Gary Alan (2002), *Shared Fantasy: Role Playing Games as Social Worlds*, University Of Chicago Press, Chicago, IL.

Gamma, Erick, Ralph Johnson, Jonh Vlissides & Richard Helm (2000), *Padrões de Projeto*, 1ª edição, Bookman.

Ganssle, Jack G. (2007), *Embedded Systems*, Elsevier Inc., Barlington, MA.

Garlan, David, Daniel P. Siewiorek & Peter Steenkiste (2002), 'Project aura: Toward distraction-free pervasive computing', *IEEE Pervasive Computing* **1**, 22–31.

Google (2009), 'Orkut', http://www.orkut.com.

Grammenos, Dimitris, Anthony Savidis & Constantine Stephanidis (2009), 'Designing universally accessible games', *ACM Computers in Entertainment* **7**(1), 22–31.

Greenhalgh, Chris (1999), *Large Scale Collaborative Virtual Environments*, Springer.

Grosso, William (2002), *Java RMI*, O'Reilly Media, Inc.

Grubb, Penny & Armstrong A. Takang (2003), *Software Maintenance: Concepts And Pract*, 6ª edição, World Scientific Publishing Company.

Haller, Michael, Bruce Thomas & Mark Billinghurst (2007), *Emerging technologies of augmented reality: interfaces and design*, 1ª edição, Idea Group Pub.

Han, JungHyun, Hoh Peter In & Jong-Sik Woo (2005), Towards situation-aware cross-platform ubi-game development, *em* 'Proceedings of Pervasive2005', pp. 8–13.

Hartley, Richard & Andrew Zisserman (2003), *Multiple View Geometry in Computer Vision*, 2ª edição, Cambridge University Press.

Jelinek, Frederick (1998), *Statistical Methods for Speech Recognition*, 1ª edição, The MIT Press.

Jones, Randolph M. (2000), 'Design and implementation of computer games: a capstone course for undergraduate computer science education', *ACM SIGCSE Bulletin* **32**(1), 260–264.

Lang, Ulrich & Rudolf Schreiner (2002), *Developing secure distributed systems with CORBA*, Artech House.

Laurent, Simon St., Joe Johnston & Edd Dumbill (2001), *Programming web services with XML-RPC*, O'Reilly Media, Inc.

Lekakos, George, Konstantinos Chorianopoulos & Georgios Doukidis (2000), *Interactive Digital Television: Technologies and applications*, Vol. 13, IGI Publishing, Hershey,pA.

Lindt, Irma, Jan Ohlenburg, Uta Pankoke-Babatz, Sabiha Ghellal, Leif Oppermann & Matt Adams (2005), Designing cross media games, *em* 'Proceedings of Pervasive'05', Munich, Germany.

Makena Technologies, Inc. (2009), 'There', http://www.there.com/.

Morningstar, Chip & Randy Farmer (1991), The lessons of lucasfilm's habitat, *em* M.Benedikt, ed., 'Proceedings of Cyberspace: First Steps', pp. 273–302.

Niemelä, Eila & Teemu Vaskivuo (2004), Agile middleware of pervasive computing environments, *em* 'PerCom Workshops', pp. 192–197.

Nolan, Stuat (2002), IDTV gamers: The emergence of a new community?, *em* F.Mäyrä, ed., 'Proceedings of Computer Games and Digital Cultures Conference', Tampere, pp. 109–113.

Ohta, Yuichi & Hideyuki Tamura (1999), *Mixed Reality: Merging Real and Virtual Worlds*, Springer.

Peiris, Chris, Dennis Mulder, Shawn Cicoria, Amit Bahree & Nishith Pathak (2007), *Pro WCF: practical Microsoft SOA implementation*, Apress.

Piccolo, Lara Schibelsky, Amanda Meincke Melo & Maria Cecilia Calani Baranauskas (2007), *Accessibility and Interactive TV: Design Recommendations for the Brazilian Scenario*, Springer, capítulo Human-Computer Interaction -INTERACT 2007.

Preiser, Wolfgang F. E. & Elaine Ostroff (2001), *Universal design handbook*, McGraw-Hill handbooks.

Pressman, Roger (2006), *Engenharia de Software*, 6ª edição, McGraw Hill.

Regenbrecht, H., T. Lum, P. Kohler, C. Ott, M. Wagner, W. Wilke & E. Mueller (2004), 'Using augmented virtuality for remote collaboration', *Presence: Teleoper. Virtual Environ.* **13**(3), 338–354.

Rellermeyer, Jan S., Oriana Riva & Gustavo Alonso (2008), Alfredo: An architecture for flexible interaction with electronic devices, *em* 'Middleware', pp. 22–41.

Rolim, Cledja, Alejandro C. Frery, Eliana Almeida, Evandro Costa & Luiz M. G. Gonçalves (2007), 'Enhancing the experience of 3d virtual worlds with a cartographic generalization approach', *Visual Computuer* **23**(1), 409–418.

Schollmeier, Rudiger (2002), 'A definition of peer-to-peer networking for the classification of peer-to-peer architetures and applications', *IEEE Internet Computing* .

Schröder, Marc (2001), Emotional speech synthesis: A review, *em* '7th European Conference on Speech Communication and Technology', pp. 561–564.

Sproat, Richard (1997), *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*, 1ª edição, Springer.

Stephenson, Neal (2007), *Snow Crash (SFBC 50th Anniversary Collection)*, SFBC.

Sutherland, Ivan (1965), The ultimate display, New York, EUA, pp. 506–508.

Sutherland, Ivan (1968), A head-mounted three dimensional display, *em* 'Fall Joint Computer Conference', Washington, EUA, pp. 757–764.

Szymanski, Boleslaw K. & Bulent Yener (2005), *Advances in Pervasive Computing and Networking*, 1ª edição, Springer.

Tanenbaum, Andrew S. & Maarten van Steen (2001), *Distributed Systems: Principles and Paradigms*, 1ª edição, Prentice-Hall.

Tkachenko, Dmitry, Nickolay Kornet & Alan Kaplan (2004), Convergence of IDTV and home network platforms, *em* 'Proceedings of IEEE Consumer Communications and Networking Conference', pp. 624–626.

Tran, Frederic Dang, Marina Deslaugiers, Anne Gerodolle, Laurent Hazard & Nicolas Rivierre (2002), An open middleware for large-scale networked virtual environments, *em* 'Proceedings of the IEEE Virtual Reality'.

Trinta, Fernando, Carlos Ferraz & Geber Ramalho (2006), Middleware services for pervasive multiplatform networked games, *em* 'Proceedings of Netgames'06', Singapore.

Vinoski, Steve (2004), An overview of middleware, *em* 'Ada-Europe', pp. 35–51.

Volter, Markus, Michael Kircher & Uwe Zdun (2005), *Remoting Patterns - Foundations of Enterprise, Internet and Realtime Distributed Object Middleware*, 1ª edição, John Wiley and Sons Ltd.

Weiser, Mark (1991*a*), 'The computer for the twenty-first century', *Scientific American* **3**(265), 94–102.

Weiser, Mark (1991*b*), 'The computer for the twenty-first century', *Scientific American* pp. 94–10.

Whitmire, Scott A. (1997), *Object-oriented design measurement*, 1ª edição, Wiley Computer Pub.

Wolf, Mark J. P. (2008), *The Video Game Explosion: A History from Pong to Playstation and Beyond*, Greenwood Publishing Group, Westport, CT.

Yatsu, Keisuke & Yoshitaka Shibata (2009), A middleware system to realize virtual reality on tele-immersion environment, *em* 'International Conference on Advanced Information Networking and Applications Workshops'.