UNIVERSIDADE DO VALE DO RIO DOS SINOS
CIÊNCIAS EXATAS E TECNOLÓGICAS
PROGRAMA INTERDISCIPLINAR DE PÓS-GRADUAÇÃO EM
COMPUTAÇÃO APLICADA

Cícero Raupp Rolim

HLS: Um modelo para suporte à sistemas de localização no Holoparadigma

São Leopoldo
2007

Cícero Raupp Rolim

HLS: Um modelo para suporte à sistemas de localização no Holoparadigma

Dissertação apresentada à Universidade do Vale do Rio dos Sinos como requisito parcial para obtenção do título de Mestre em Computação Aplicada.

Orientador: Prof. Dr. Jorge Luís Victória Barbosa

São Leopoldo

2007

CIP — CATALOGAÇÃO NA PUBLICAÇÃO

Rolim, Cícero Raupp

HLS: Um modelo para suporte à sistemas de localização no Holoparadigma / por Cícero Raupp Rolim. — São Leopoldo: Ciências Exatas e Tecnológicas da UNISINOS, 2006.

86 f.: il.

Dissertação (mestrado) — Universidade do Vale do Rio dos Sinos. Ciências Exatas e Tecnológicas. Programa Interdisciplinar de Pós-Graduação em Computação Aplicada, São Leopoldo, BR–RS, 2006. Orientador: Barbosa, Jorge Luis Victória.

1. Computação móvel. 2. Computação ubíqua. 3. Sistemas de localização. 4. Aplicações sensíveis à localização. I. Barbosa, Jorge Luis Victória. II. Título.

UNIVERSIDADE DO VALE DO RIO DOS SINOS

Reitor: Dr. Marcelo Fernandes de Aquino

Diretora da Unidade de Pós-Graduação e Pesquisa: Profª. Drª. Ione Bentz

Coordenador do PIPCA: Prof. Dr. Arthur Tórgo Gómez

"Embora ninguém possa voltar atrás e fazer um novo começo, qualquer um pode começar agora e fazer um novo fim."

Chico Xavier

Agradecimentos

Ao meu orientador Prof. Dr. Jorge Luis Victória Barbosa pela paciência, amizade e motivação durante o curso.

Ao Prof. Dr. Luciano Paschoal Gaspary pela amizade e atenção quando procurei antes de entrar no mestrado.

A minha mãe Maria Madalena Munari Raupp Rolim pela paciência, carinho e amor. Obrigado por tudo!

A minha sobrinha Júlia Raupp Rolim dos Santos.

A minha namorada Carolina Ramos, pela companhia, amizade, amor e pelos momentos que seriam seus.

Aos meus amigos e colegas do MobiLab Gustavo Lermen, Darci Levis, Nelson Sonntag, Rodrigo Hahn, Solon Rabello, Fabiane Dillemburg, Renato Costa, Dario Franz, Fernando Caprio Júnior e Daniel Bonatto.

Ao meu grande amigo Gustavo Bisognin, afinal, não é fácil aguentar o "Baboo" todos os dias durante dois anos, na ida e volta para a Unisinos.

A todos meus colegas do curso: na presença de vocês sempre me senti acolhido e confiante! Aprendi que grande amizades podem ser realizadas inclusive sob os momentos de maior pressão. Sempre lembrarei de vocês!

A Hewlett-Packard Computadores (HP) pela bolsa de mestrado e incentivo financeiro.

Resumo

O crescimento do poder computacional dos dispositivos portáteis como PDAs, handhelds e notebooks é uma realidade na última década. Paralelamente, as redes sem fio (por exemplo Wi-Fi e *bluetooth*), tiveram um crescimento vinculado a estes equipamentos, facilitando a comunicação e troca de informações entre os mesmos. Nesse escopo surgiu a computação ubíqua. No escopo da computação ubíqua, as aplicações devem ser sensíveis à rede, recursos, localização física e contexto, ou seja, podem ter seu comportamento alterado durante sua execução, devido à mobilidade constante dos dispositivos móveis. Neste cenário o Holoparadigma apresenta-se como uma proposta de solução para a manipulação das questões associadas à mobilidade, já que possui um modelo de programação intuitivo, tornando possível a modelagem de ambientes utilizando as suas abstrações.

Este trabalho apresenta o HLS, um modelo para desenvolvimento de aplicações sensíveis à localização utilizando o ambiente do Holoparadigma. O HLS é um modelo que inclui um servidor de localização de dispositivos móveis, um módulo que é integrado ao ambiente de execução do Holoparadigma para alteração do estado de execução das aplicações, e uma ferramenta para administração de ambientes com servidor de localização baseado em árvores de contexto.

Palavras-chave: Computação móvel, Computação ubíqua, Sistemas de localização, Aplicações sensíveis à localização.

TITLE: "LOCATION SYSTEM FOR HOLOPARADIGM"

Abstract

The growing computational power of mobile devices like PDAs, handhelds and notebooks is a reality in the last decade. At the same time, Wi-Fi networks (IEEE 802.11 and bluetooth) technology both shared a growth tendency alongside mobile devices, making easier to share information between them. These advances made ubiquitous computing possible. In the context of ubiquitous computing, applications should be aware of network, resources, physical location and context, in other words, their behavior can be changed while still in execution, due to mobile devices' inherent and constant mobility. In this scenario, the Holoparadigm presents itself as a possible solution to mobility related questions. It has an intuitive programming model, which eases ambient modeling using it's abstracts.

This paper presents HLS, a location-aware application development model, using the Holoparadigm environment. The HLS model includes a mobile devices location server, a module which is integrated to the Holoparadigm execution environment to manage changes in the applications execution state, and a tool for managing environments associated with the location server, based in context trees.

Keywords: Mobile computing, ubiquitous computing, location systens, location-aware applications.

Sumário

Resumo **5**

Abstract **6**

Lista de Abreviaturas **10**

Lista de Figuras **12**

Lista de Tabelas **14**

1 Introdução **15**

1.1 Contextualização 15

1.2 Definição do problema 17

1.3 Objetivos 17

1.4 Metodologia 18

1.5 Organização do texto 19

2 Sistemas de Localização **20**

2.1 Taxonomia 20

2.2 GPS 21

2.3 Infra-vermelho 23

2.4 Ultra-som 23

2.5 RFID 24

2.6 IEEE 802.11 25

2.7 Conclusões 26

3 Sistemas Sensíveis à Localização **27**

3.1 Aura 27

3.2 Gaia 30
3.3 one.world 31
3.4 PLACE 32
3.5 Conclusões 33

4 Holoparadigma **35**
4.1 Principais Conceitos 35
4.2 Holo Tree 37
4.3 HNS e HVM 38
4.4 Taxonomia de entes para mobilidade 40
4.5 Conclusões 43

5 Modelo HLS **44**
5.1 Arquitetura da Solução 44
5.2 Servidor de Localização 45
5.3 *Context Changer* 48
5.4 *Holo Tree View* 50
5.5 Conclusões 51

6 Aspectos de Implementação **53**
6.1 SELIC 53
6.1.1 Arquitetura do SELIC 53
6.1.2 Precisão 55
6.1.3 Consumo de Energia 58
6.2 *Context Changer* 61
6.3 *Holo Tree View* 63
6.4 Conclusões 65

7 Aplicações **67**
7.1 LOCAL 67
7.2 Agenda *Pervasiva* 69
7.3 Controle de segurança 71
7.4 Conclusões 74

8 Considerações Finais **75**

8.1 Conclusões 75

8.2 Contribuições 76

8.3 Trabalhos Futuros 77

Bibliografia **79**

Lista de Abreviaturas

AP Access Point

API Application Programming Interface

CC Context Changer

CASE Computer Aided Systems Engineering

DAS Device Access Service

DSM Distributed Shared Memory

GPS Global Positioning System

GUI Graphical User Interface

HNI Holo Native Interface

HNS Holo Naming System

HOLO Holoparadigma

HS History Server

HTV Holo Tree View

HVM Holo Virtual Machine

HTTP HyperText Transfer Protocol

IEEE Institute of Electrical and Electronics Engineers

IP Internet Protocol

JVM Java Virtual Machine

KS Knownledge Source

LBS Location Based Services

NAT Network Address Translation

PB Physical Beings

PDA Personal Digital Assistent

PHOLO Pervasive Holoparadigm

PSB Physical Static Being

PDB Physical Dynamic Being

RFID Radio-Frequency Identification

RSSI Received Signal Strength Indicator

SIA Service Interface Architecture

SNMP Simple Network Management Protocol

SOAP Simple Object Access Protocol

SSID Service Set Identifier

SQL Structured Query Language

UDP User Datagram Protocol

VB Virtual Beings

VDB Virtual Dynamic Being

VM Virtual Machine

VS Visual Studio

VSB Virtual Static Being

XML Extended Modeling Language

Lista de Figuras

2.1 Órbita dos satélites responsáveis pela eficiência e cobertura da tecnologia GPS 22

3.1 Entidades e relacionamento entre entidades no Aura [1] 28
3.2 Arquitetura do *Service Interface Architecture* [1] 29
3.3 Arquitetura do *MiddleWhere* [2] 32
3.4 Tela do sistema PLACE utilizando GPS 33

4.1 Entes Elementar, Composto e Composição em 3 níveis 36
4.2 Exemplo de mobilidade no Holoparadigma 37
4.3 Organização hierarquica no Holoparadigma - *HoloTree* 38
4.4 Figura apresentando funcionamento do HNS [3] 39
4.5 Tipos de Entes no Holoparadigma 41
4.6 Exemplo da *HoloTree* para a aplicação "*MobiLab Pervasive Learning*" 42

5.1 Arquitetura do *Holo Locating System* 45
5.2 Arquitetura do SELIC 46
5.3 Apresentação do módulo *Context Changer* 49
5.4 Exemplo da situação inicial de uma aplicação Holo e o SELIC 49
5.5 Exemplo da de uma mobilidade sobre o exemplo da Figura 5.4 50

6.1 Apresentação dos *Web Services* do SELIC 55
6.2 Tela de autenticação no sistema SELIC (cliente) 56
6.3 *Engine* do SELIC realizando um cálculo de posicionamento 56
6.4 Distribuição dos pontos de acesso no segundo andar do prédio 6B da Unisinos 57
6.5 Precisão do SELIC no segundo andar do prédio 6B da Unisinos . . . 58

6.6 Gráfico apresentando o consumo de bateria apresentado na Tabela 6.2 61
6.7 Código para a mudança da *DHoloTree* pelo *Context Changer* 62
6.8 Exemplo de posicionamento de dispositivos 63
6.9 Exemplo de saída da *Holo Tree View* para a situação da Figura 6.8 . 64
6.10 Exemplo de mudança de contextos no mundo físico 64
6.11 Exemplo de saída da HTV para a Figura 6.10 65

7.1 Arquitetura do LOCAL 68
7.2 Arquitetura da aplicação Agenda *Pervasiva* 72
7.3 Código fonte do *HoloGuard* 73

Lista de Tabelas

6.1 Descrição dos equipamentos 59
6.2 Consumo de bateria com o cliente do SELIC 60

7.1 Simulação de uma aula usando o LOCAL 70

Capítulo 1

Introdução

1.1 Contextualização

O crescimento do poder computacional dos dispositivos portáteis como PDAs, *handhelds* e *notebooks* é uma realidade na última década. Paralelamente as redes sem fio (por exemplo, Wi-Fi e *bluetooth*) tiveram um crescimento vinculado a estes equipamentos, facilitando a comunicação e troca de informações entre os mesmos.

A partir desta evolução as pessoas estão tendo cada vez mais seu dia-a-dia atrelado a dispositivos computacionais de pequeno porte, popularizando a pesquisa e busca de novas aplicações, que permitam uma integração entre dispositivos e locais físicos. A primeira visão computacional deste mundo foi realizada por Mark Weiser, em 1991 [4]. Mark descreveu um mundo em que os dispositivos computacionais interagiam naturalmente com as pessoas, que passavam a fazer parte deste sistema. Desta visão surgiu o termo "computação ubíqua".

No escopo de computação ubíqua, surgiram diversas propostas para fornecer um arcabouço para desenvolvimento de aplicações que sejam sensíveis à rede (*network-aware*), recursos (*resource-aware*), localização física (*location-aware*) e contexto (*context-aware*). Estes são os requisitos fundamentais para uma aplicação ubíqua [5]. Estes requisitos favorecem o surgimento de um novo paradigma, que seja capaz de tratar de forma nativa a mobilidade que estes dispositivos oferecem a seus usuários. Neste cenário o Holoparadigma [6, 7, 8, 9], de forma abreviada Holo, apresenta-se como uma proposta de solução para a manipulação das questões associadas à mobilidade, já que possui um modelo de programação intuitivo, facilitando a modelagem de ambientes ubíquos. Atualmente o Holoparadigma conta

com a Hololinguagem e um ambiente de execução, que permitem aplicar os princípios do paradigma. O desenvolvimento de novos recursos para a Hololinguagem e o ambiente de execução, estão sendo realizados no MobiLab [10]. O MobiLab é o laboratório de pesquisa e desenvolvimento em computação móvel da Unisinos, e suas principais áreas de pesquisa são: computação móvel, computação ubíqua, educação ubíqua, jogos de computadores e linguagens de programação.

A localização física consiste no local onde encontra-se o equipamento. Oferecer o suporte à localização física para aplicações ubíquas é um desafio que possui uma atenção especial por diversas plataformas tais como Aura [11, 12, 13], Gaia [14, 15, 16], one.world [17] e PLACE [18, 19, 20]. Com base na localização de um equipamento, as aplicações mudam seu estado, com a finalidade de oferecer serviços e conteúdo relacionado com a localização. A adaptação automática da aplicação em relação à localização física é uma abordagem que merece destaque, porque permite uma menor intervenção do usuário junto ao programa. Este tipo de aplicação é denominado *Location Based Services* (LBS) [21, 22, 23, 24, 25]. A grande maioria dos *frameworks* provê a informação de posicionamento automaticamente ao programador, permitindo que o foco do desenvolvimento esteja na própria aplicação, e não no modo como a informação de localização será adquirida.

O suporte a aplicações sensíveis à localização traz uma série de novos desafios que devem ser analisados, como o sistema de localização a ser utilizado, a precisão do sistema de localização, o armazenamento do histórico de locais onde um equipamento foi encontrado (*tracking*), o comportamento da aplicação perante a mobilidade, busca por novos locais de posicionamento, a administração de ambientes entre outros.

Considerando a necessidade de aquisição de informações sobre o posicionamento de dispositivos móveis, é importante mencionar que uma variedade de serviços adicionais devem estar disponíveis as aplicações e administradores. Atualmente existem implementações que prestam serviços de localização, porém grande parte utiliza apenas um sistema de localização para obter o posicionamento, restringindo a implementação num grande ambiente. Isso ocorre porque alguns sistemas de localização sofrem atenuações quando o equipamento está dentro de um prédio, como por exemplo o GPS.

Um recurso que pode ser vinculado ao posicionamento físico de equipamentos

é o *tracking*. O *tracking* é o recurso que permite armazenar em um banco de dados, o histórico de locais que um equipamento visitou. Pode-se utilizar o *tracking* para rastrear o caminho de um dispositivo no sistema e/ou detecção de super lotação em ambientes. Uma utilização de *tracking* que merece destaque é a análise de recursos que podem ser de interesse do usuário, baseado no seu perfil de deslocamento.

1.2 Definição do problema

No ambiente Holo a unidade básica de modelagem e programação pode representar qualquer entidade. Neste contexto é necessária a definição de uma taxonomia para classificar esta unidade conforme o seu perfil de mobilidade. Após definir a taxonomia é necessário analisar o ambiente de execução do Holoparadigma, que é composto de máquinas virtuais (HoloVM) [26] e o HNS [3, 27]. O HNS é um sistema que permite integrar todas as máquinas virtuais que estão executando código Holo. Atualmente o HNS não possui suporte a informação de posicionamento de equipamentos. Ao integrar um servidor de localização com o HNS é possível disponibilizar informações de posicionamento de equipamentos para o ambiente distribuído Holo, e assim utilizar este recurso como forma de alterar o estado da aplicação Holo em tempo de execução, ou seja, fornecer o suporte à aplicações sensíveis à localização no Holoparadigma.

Em lugares onde os dispositivos tem sua localização disponível, é possível utilizar esta informação como meio de gerenciar ambientes. A verificação da quantidade de equipamentos em uma sala, ou a disponibilidade de um ambiente são exemplos de funcionalidades que podem ser gerenciadas. Este cenário pode ser explorado através de uma aplicação de monitoramento, que apresente as informações de posicionamento de dispositivos através de árvores de contexto.

A questão de pesquisa deste trabalho é como integrar sistemas de localização no Holoparadigma e no seu ambiente de execução distribuído.

1.3 Objetivos

Este trabalho tem por objetivo especificar, implementar e avaliar uma solução que integre o suporte de aplicações sensíveis à localização no Holoparadigma. Os

objetivos específicos deste trabalho são:

- Revisar as soluções existentes;
- Definir uma arquitetura para integração de aplicações sensíveis à localização no Holoparadigma;
- Criar um servidor de localização, com suporte a múltiplos sistemas de localização, que tenha o recurso de *tracking* e disponibilize uma interface de comunicação com aplicações através de *Web Services* [28];
- Criar uma taxonomia para classificação da unidade básica de modelagem do Holoparadigma, de acordo com o seu perfil de mobilidade;
- Criar um módulo integrado ao HNS para a obtenção de informações de posicionamento junto ao servidor de localização, e alterar o estado da aplicação em tempo de execução;
- Criar uma ferramenta para o monitoramento de dispositivos identificados pelo servidor de localização e HNS;
- Avaliar a solução proposta.

1.4 Metodologia

O primeiro passo foi o estudo dos sistemas de localização e das soluções que integrem à localização física de dispositivos com ambientes ubíquos. O principal objetivo neste estudo foi verificar como são realizadas as consultas sobre informações de posicionamento e como estas são disponibilizadas ao sistema.

O passo seguinte foi fazer um estudo sobre o Holoparadigma e suas abstrações. Neste estudo o objetivo foi entender quais tipos de aplicações sensíveis ao contexto podem ser desenvolvidas utilizando este paradigma.

Foi criado um servidor de localização, capaz de disponibilizar a informação de posicionamento de equipamentos. Esta informação pode ser adquirida de diversos sistemas de localização. Esta tarefa teve como objetivos: (1) conhecer a precisão de localização do sistema IEEE 802.11 para o segundo andar do prédio 6B da Unisinos;

(2) verificar a disponibilidade das informações através do padrão *Web Services*; (3) testar a informação de *tracking* de dispositivos.

Depois da implementação do servidor de localização, foi desenvolvido um módulo, integrado ao HNS que permite receber as informações de posicionamento dos dispositivos do servidor de localização, e que realiza a alteração do estado da aplicação Holo. Com esse módulo foi possível a execução de programas sensíveis à localização no ambiente Holo.

Na etapa seguinte foi desenvolvida uma ferramenta que permite a administração de ambientes físicos, que utilize como fonte de informações o servidor de localização e o HNS. O principal objetivo desta ferramenta é permitir uma visualização do posicionamento dos equipamentos e ambientes. A saída desta ferramenta é uma árvore de contextos, para manter o padrão apresentado pelo Holoparadigma.

Por último, para validar o ambiente, foram desenvolvidas aplicações sensíveis à localização, que utilizaram os serviços disponíveis no ambiente Holo durante sua execução.

1.5 Organização do texto

O restante do trabalho está organizado como segue. No Capítulo 2 é realizada uma revisão sobre os sistemas de localização mais utilizados. O Capítulo 3 apresenta um estudo sobre sistemas sensíveis à localização. O Capítulo 4 discute o Holoparadigma e as tecnologias em uso no mesmo. O Capítulo 5 apresenta o modelo de aplicações sensíveis à localização proposto para o Holoparadigma. O Capítulo 6 aborda a estratégia escolhida para a prototipação do modelo. No Capítulo 7 são apresentadas aplicações que utilizam o modelo proposto. No Capítulo 8 encontram-se as conclusões e os trabalhos futuros.

Capítulo 2

Sistemas de Localização

Neste capítulo será feita uma revisão sobre os sistemas, que são direcionados para a localização de equipamentos em ambientes. Esta revisão tem por objetivo identificar quais são as principais características dos sistemas, e desta forma acumular o conhecimento necessário para propor um servidor que integre diversas tecnologias de localização.

2.1 Taxonomia

Um dos tópicos vitais para a computação ubíqua é a necessidade das aplicações terem acesso a informações sobre a localização de dispositivos. A localização é o local físico ou simbólico onde está um equipamento.

A informação de localização pode ser dividida em dois tipos [29]:

- *Física:* é aquela em que é possível obter a coordenada geográfica do equipamento, baseada em pontos conhecidos e pré-determinados, e geralmente é adquirida utilizando um dispositivo de GPS;
- *Simbólica:* é aquela em que um significado abstrato pode representar um contexto, por exemplo "Unisinos", "Prédio 6B" ou "Sala 212".

Existem três principais técnicas para determinar a localização [30, 31]:

- Triangulação: pode ser realizada via lateração, onde a posição de um ponto é obtida usando as distâncias entre o ponto e outros pontos cuja posição é conhecida, ou angulação, que é semelhante ao usado na lateração. No lugar

dos valores da distância são usados os ângulos entre pontos de referência e o ponto a ser posicionado;

- Proximidade: a técnica de sensoriamento baseada em proximidade obtém a posição de um determinado objeto ou pessoa através da "detecção de proximidade" em relação a um ponto cuja localização é conhecida;
- Análise de Cenário: sistemas de localização através de análise de cenário baseiam-se na análise de características de dados capturados do ambiente. A análise de cenário pode fornecer informações tanto sobre os objetos observados (caso a posição do observador seja conhecida) como sobre a localização do observador (caso os dados obtidos indiquem a presença de pontos de referência cuja posição é conhecida).

Os servidores de localização provêem serviços às aplicações. As aplicações mantêm seu comportamento baseadas nas informações adquiridas junto a estes serviços, e realizam consultas ou recebem avisos quando algum disposito trocou de contexto. Esses servidores geralmente implementam duas ou mais técnicas para a localização de dispositivos. Isto é utilizado quando procura-se ter uma melhor precisão na localização, ou quando o sistema de localização é implantado em um grande centro. Algumas técnicas podem sofrer interferência de alguns fatores, como por exemplo estruturas de concreto, lagos, forno microondas entre outros.

Nas próximas seções serão apresentados os sistemas de localização mais utilizados.

2.2 GPS

O *Global Positioning System* é possivelmente o sistema de localização mais conhecido e utilizado. O GPS é um sistema de navegação, baseado em lateração, que fornece à localização física, apoiado por uma rede de satélites colocados em órbita pelo Departamento de Defesa dos Estados Unidos e que possuem uma cobertura confiável e irrestrita (Figura 2.1) [32]. Originalmente o sistema foi destinado à aplicações militares, entretanto, na década de 80, o governo americano tornou o sistema disponível para uso civil.

Figura 2.1 – Órbita dos satélites responsáveis pela eficiência e cobertura da tecnologia GPS

Receptores GPS calculam sua posição baseando-se na informação recebida de uma rede de satélites em órbita. O sistema usa o tempo de propagação dos sinais de rádio dos satélites até os receptores para fazer esse cálculo. Todos os satélites que constituem a infra-estrutura do GPS possuem relógios sincronizados, mantendo uma precisão de 1 segundo em 1013 segundos. Cada receptor resolve um sistema de quatro equações com quatro incógnitas (coordenadas espaciais x, y, z e tempo de transmissão) usando os intervalos entre a recepção dos sinais de pelo menos quatro satélites.

A precisão da localização é entre 1 e 5 metros. Os *receivers* estão com valores cada vez mais acessíveis, e atualmente é possível encontrar dispositivos como PDAs, *laptops* e celulares equipados com suporte a esta tecnologia. Como principal desvantagem cita-se a dificuldade em utilizar o sistema em ambientes fechados, devido à baixa freqüência dos sinais. Alguns *receivers* possuem o problema de continuar informando a última coordenada localizada, após perderem a visada aos satélites [33].

A queda de precisão de *receivers* GPS em ambientes fechados, motivou a empresa QUALCOMM a desenvolver um produto para a telefonia celular, chamado gpsOne [34, 35]. O gpsOne integra dois sistemas de localização, o GPS e a triangulação de antenas de celular para obter uma melhor precisão.

O gpsOne é totalmente integrado no chipset QUALCOMM. Os aparelhos celulares não necessitam a instalação de nenhum hardware ou software adicional.

2.3 Infra-vermelho

Infravermelho é a parte do espectro eletromagnético com freqüência entre 300GHz e 300THz. Uma propriedade importante é que sinais infravermelho não são capazes de atravessar paredes como sinais de rádio. Outra propriedade interessante do infravermelho é que o mesmo não é visível pelo olho humano [29].

A tecnologia infravermelho tem sido usada em um grande número de aplicações comerciais, desde controle remotos até equipamentos de visão noturna. Por já ser explorada comercialmente, a tecnologia de infravermelho é de baixo custo e altamente disponível para o desenvolvimento de novas aplicações.

O *Active Badge Location System* [36] foi desenvolvido no *AT&T Cambridge*. O *Badge*, utilizado por usuários ou objetos a serem localizados, emite um sinal infravermelho com um identificador global único a cada 15 segundos ou sob demanda. Um servidor na rede lê os dados dos sensores infravermelhos fixos no ambiente, agrega-os, e provê uma *API* para acesso aos dados.

2.4 Ultra-som

Ultra-som é a parte do espectro eletromagnético com freqüência acima de 20KHz. Semelhante ao infra-vermelho, o ouvido humano não é capaz de escutar sons nessa freqüência. É largamente utilizado na indústria e na medicina, como exemplo pode-se citar aparelhos de ultra-sonografia.

A utilização de ultra-som requer uma complexa infra-estrutura em todo o ambiente de interesse, sendo particularmente sensível à posição desses sensores. Desta forma, falta de escalabilidade, dificuldade de implantação e custos elevados são as principais desvantagens desta tecnologia.

O *Active Bat* [30] foi desenvolvido no *AT&T Cambridge*, e utiliza a técnica de multilateração [37] para prover uma localização mais precisa do que o sistema *Active Badge*.

Usuários e objetos utilizam crachás, que emitem um pulso ultra-sônico captado

por uma grade de receptores montados no teto do ambiente e, simultaneamente, um sinal de rádio freqüência. Cada receptor calcula a diferença entre o intervalo de tempo do sinal de rádio freqüência e do pulso ultra-sônico para obter a distância ao crachá.

O *Cricket Location System* [38, 39], utiliza emissores ultra-som no ambiente e receptores de baixo custo embutidos nos objetos a serem localizados. Essa estratégia exige ao dispositivo calcular sua própria informação de localização através de multilateração. Semelhante ao sistema *Active Bat*, o *Cricket* utiliza pulsos ultra-sônicos e sinais de rádio freqüência.

Entretanto, o sistema não necessita de uma grade de sensores de teto com posições fixas como no *Active Bat* porque o receptor, no dispositivo móvel, é o responsável por calcular sua própria localização. O sistema *Cricket* representa um compromisso entre precisão e baixo custo em hardware quando comparado ao sistema *Active Bat*.

2.5 RFID

O acrônimo RFID expande-se para a expressão inglesa *Radio-Frequency IDentification*, que em português significa identificação por rádio frequência. O RFID é uma poderosa e versátil tecnologia para identificar, rastrear e gerenciar uma enorme gama de produtos, documentos, animais e indivíduos, sem contato e sem a necessidade de um campo visual [40].

O RFID é uma tecnologia de identificação que utiliza a radiofreqüência para capturar os dados e não a luz como no caso do código de barras, com isso a tecnologia de RFID permite que um *tag* seja lido sem a necessidade de campo visual, através de objetos tais como madeira, plástico, papel etc. Essa comunicação é feita, utilizando dois componentes: o *transponder* ou *RF tag* (ou simplesmente *tag*) e um leitor com antena, que pode também ser gravador caso seja necessário escrever novos dados no *chip* do *transponder*. Quando aproxima-se um *tag*, o campo do leitor alimenta o *tag*, que transmite dados da sua memória para o leitor e vice-versa, no caso de um *tag* de leitura/escrita.

As *tags* podem ser classificadas em duas categorias:

- Passivas: São *tags* que operam sem necessidade de bateria. Refletem o sinal

transmitido para si pelos leitores. Elas são utilizadas para substituição do sistema de código de barras. Essas *tags* são mais efetivas e tem custo mais baixo que *tags* ativas, oferecendo uma vida operacional virtualmente ilimitada. Entretanto, sua área de leitura é limitada;

- Ativas: *tags* ativas possuem um transmissor de rádio e uma bateria. Possuem uma área de leitura maior que as *tags* passivas. Elas também oferecem a durabilidade necessária para a identificação permanente de certos produtos.

A tecnologia RFID não foi projetada para detecção de posicionamento, mas começou a ser utilizada como um sistema de localização com uma precisão razoável (dependendo do número de leitores) por vários sistemas de localização. Uma outra técnica para utilizar o RFID é integrando o *transponder* no dispositivo portátil, e colocar as *tags* para representar as regiões. Neste modo, quando o *transponder* detecta uma *tag*, ele realiza uma consulta ao servidor de localização para verificar a área que se encontra atualmente. Até o momento o custo desta solução é muito alto, e são poucos os dispositivos que já vêm de fábrica com o equipamento.

LANDMARC (*Location Identification based on Dynamic Active RFID Calibration*) é um sistema para localização de dispositivos móveis, que estão em lugares cobertos, baseado na tecnologia RFID [40]. O objetivo do projeto é utilizar leitores de RFID em áreas fechadas, colocando *tags RFID* nos equipamentos. Cada leitor é responsável por uma determinada área, e sua precisão é baseada na potência de sua antena e no número de leitores vizinhos. A área de cobertura de uma antena de um leitor é aproximadamente 50 metros.

O desempenho deste sistema é determinado pelo número de leitores, sua localização e sua potência. Esses leitores devem ser distribuídos uniformemente nos ambientes onde serão implementados. Para a localização existe um servidor central que fornece a localização de cada *tag*, através de uma interface de comunicação com as aplicações.

2.6 IEEE 802.11

Sistemas de localização baseados no padrão IEEE 802.11 podem ser divididos em dois grupos, conforme o tipo de informação de localização que utilizam para o

cálculo do posicionamento [41, 42]:

- Sistemas que utilizam informações sobre o ponto de acesso (AP) no qual estão associados. Isto é feito pelo identificação de cada ponto de acesso, denominada SSID (*Service Set Identifier*). Como cada ponto de acesso possui um SSID único, torna-se possível distinguir este ponto de acesso de todos os outros. Esses sistemas mantêm armazenados, em uma base de dados, para cada ponto de acesso um rótulo que identifica a localização física e/ou simbólica. Apesar desta estratégia prover uma infra-estrutura para localização com baixo custo e de fácil implantação, a precisão obtida é bastante limitada, pois depende da densidade de pontos de acesso na região de interesse [43];

- Outros sistemas utilizam informações sobre a potência de sinal de pontos de acesso conhecidos para determinar a localização, principalmente a intensidade do sinal RSSI (*Received Signal Strength Indicator*). Tais sistemas podem ser divididos em dois grupos: os que usam técnicas determinísticas [44, 45], e os que usam técnicas probabilísticas [46, 47, 48] na inferência.

2.7 Conclusões

Nesta capítulo foram apresentados os principais sistemas de localização. Cada sistema possui características específicas, como técnica de localização, precisão e custo. A crescente procura por tecnologias de localização tem propiciado uma queda no preço destes sistemas, permitindo utilizar várias tecnologias para aprimorar a precisão da informação de posicionamento.

Com a disponibilidade e popularização dos equipamentos e sensores de localização, é interessante ter uma tecnologia híbrida, capaz de lidar com diferentes sistemas e protocolos. A implementação de um servidor, que suporte diversos sistemas de localização, é uma estratégia que permite fornecer informações em alto-nível as aplicações. Neste modo o foco do desenvolvimento fica no programa, e não há perda de tempo no tratamento em como a informação de posicionamento será adquirida, processada ou armazenada.

Capítulo 3

Sistemas Sensíveis à Localização

Neste capítulo será feita uma revisão sobre projetos que utilizam sistemas sensíveis à localização. Esta revisão apresenta o estado da arte de algumas soluções, e tem como objetivo fornecer uma base de conhecimento para a definição de um modelo de aplicação sensível à localização orientado ao Holoparadigma.

3.1 Aura

O Aura [11, 12, 13] é um projeto que vem sendo desenvolvido na Universidade de Carnegie Mellon. O Aura é um sistema para a computação ubíqua que tem dois objetivos principais: (1) maximizar o uso dos recursos disponíveis, tanto de processamento quanto de comunicação; (2) minimizar a distração do usuário com fatores externos a aplicação.

O conceito é de que o usuário passa a ter uma "aura pessoal". Quando este entra um novo ambiente, sua "aura" se encarrega de conseguir os recursos necessários para que o usuário execute a sua tarefa. Exemplo de tarefas: escrever um artigo, uma apresentação ou mesmo comprar uma casa, onde cada uma destas tarefas pode envolver uma grande quantidade de fontes de informação e aplicações [3].

O aura permite que sejam desenvolvidas aplicações conscientes de localização e contexto. A arquitetura para aquisição dessas informações é composta por quatro entidades [1]:

- *Devices* - gerencia informações sobre dispositivos, como computadores, *notebooks*, PDAs e impressoras;

- *People* - gerencia informações sobre os perfis dos usuários dos sistemas;
- *Networks* - gerencia informações sobre a rede, como por exemplo taxas de trasmissão, usuários conectados ou largura de banda;
- *Areas* - gerencia informações referentes ao contexto, como por exemplo objetos presentes, número de cadeiras disponíveis ou se existe algum quadro com canetas hidrocores.

As entidades *Devices*, *People*, *Networks* e *Areas* representam os componentes de um ambiente. As entidades podem ter suas informações relacionadas de forma semelhante a uma consulta SQL. Essas informações podem ser obtidas pelas aplicações através da criação de uma consulta ao *Service Interface Architecture* (SIA) [1, 49, 50], que é responsável pelo gerenciamento das informações. As consultas podem utilizar mais de uma entidade, ou seja, pode fazer relacionamentos entre as entidades para aprimorar o desempenho e precisão. Por exemplo, a lista de todos os dispositivos que estão na área "*Z*" (*Areas*), que são do tipo PDA (*Devices*) e possuem conexão por rede Wi-Fi (*Networks*). Na Figura 3.1 é apresentado o relacionamento entre as entidades.

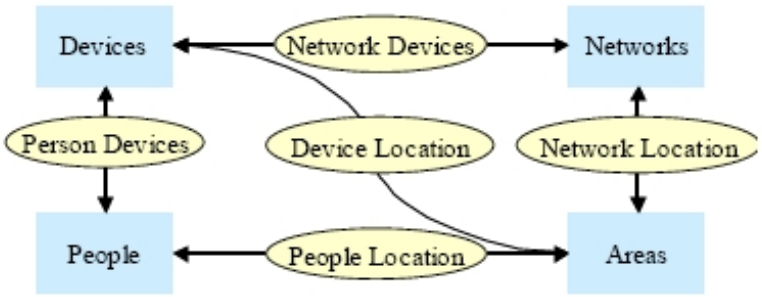

Figura 3.1 – Entidades e relacionamento entre entidades no Aura [1]

O SIA é o responsável pela aquisição e manutenção das informações de localização e contexto que serão disponibilizadas as aplicações. No caso das informações de posicionamento e contexto que na maioria do tempo serão estáticas, é utilizado um banco de dados como *background*, como por exemplo a posição de uma mesa ou computador *desktop*. As informações de localização dinâmicas são adquiridas através de consultas SNMP aos dispositivos, ou através de uma

interface CORBA [51, 52], que buscam o conteúdo nos sensores/pontos de acesso e realizam o processamento. Quando a consulta é repassada ao SIA, ela é analisada, e encaminhada para o serviço de localização. Após o serviço de localização adquirir a informação, ela é então sintetizada e repassada ao cliente (Figura 3.2).

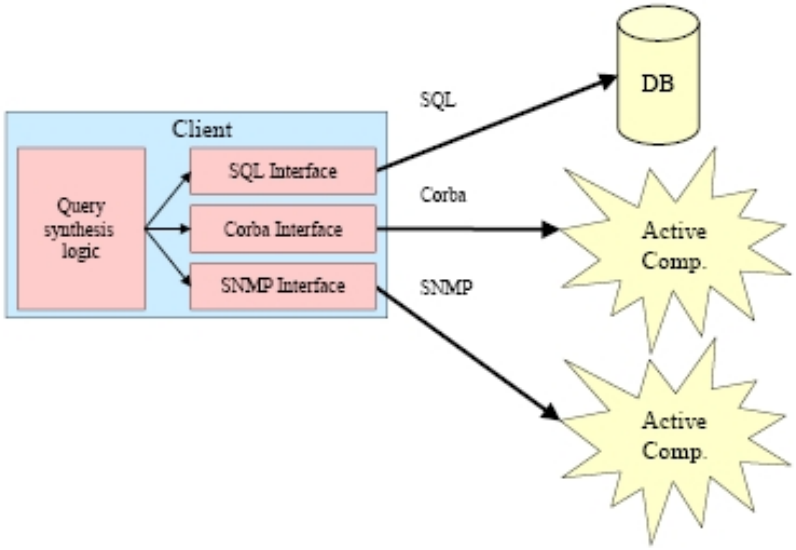

Figura 3.2 – Arquitetura do *Service Interface Architecture* [1]

O tempo para processamento e retorno da consulta também pode ser especificado. Por exemplo, se a aplicação solicitou informação sobre algum recurso de rede com prazo de execução máximo de 15 segundos, e o SIA não responde-la, a consulta será descartada. Atualmente o SIA pode suportar clientes e serviços que residem em uma grande variedade de plataformas, escritas em diferentes linguagens de programação. A interface é de tamanho reduzido, permitindo que seja integrada a plataforma de dispositivos de pequeno porte, como PDAs e celulares. Outro recurso interessante presente no SIA são os *triggers*. Os *triggers* são registrados pelas aplicações. Quando uma condição de *trigger* é satisfeira (evento), a aplicação é informada do novo estado. Os *triggers* são úteis para economia de recursos de processamento e rede, permitindo que a aplicação seja informada quando uma determinada condição estiver disponível ou tenha sido alterado o seu estado.

3.2 Gaia

O Projeto Gaia [14, 15, 16], desenvolvido na Universidade Illinois, tem como objetivo a criação de uma infra-estrutura de *middleware* distribuído que coordena serviços de software e dispositivos. Esse *middleware* pode ser utilizado em uma rede heterogênea. O Gaia possibilita a consulta por serviços, utilizando-se dos recursos existentes para acessar e manipular o contexto atual, e provê um *framework* para desenvolver aplicações com as seguintes características:

- centradas no usuário;
- conscientes do espaço físico;
- para múltiplos dispositivos.

O Gaia define um novo conceito chamado de *Active Space*. Um *Active Space* é um espaço físico coordenado por uma infra-estrutura de *software* baseada em contexto que potencializa a habilidade do usuário para interagir e configurar seu ambiente virtual e físico de forma consistente.

O suporte à localização é realizado através do serviço *MiddleWhere* [2, 53]. O *MiddleWhere* integra tecnologias de múltiplos sistemas de localização de objetos móveis (pessoas, equipamentos ou objetos físicos). Os sistemas de localização são módulos no *MiddleWhere*, permitindo que novas tecnologias sejam adicionadas ao *middleware* sem a necessidade de alterar o protocolo de comunicação com o framework. Existe um mapeamento do mundo físico para o *MiddleWhere*, permitindo que exista relacionamentos entre ambientes e objetos móveis, ou seja, é possível relacionar todos os dispositivos que estão numa sala em um determinado momento.

O serviço utiliza técnicas de probabilidade e estatística para resolver conflitos quando um ou mais sensores identificam um dispositivo em ambientes diferentes, ou seja, escolhe a posição que seja de maior vantagem para a aplicação. Por exemplo, se um equipamento é identificado numa sala por um sensor RFID (que possui uma precisão maior para ambientes fechados) e ao mesmo tempo é localizado em outra sala pela triangulação de antenas IEEE 802.11, é escolhido o RFID, porque a área de abrangência do sensor é menor, mas a precisão é mais alta.

As principais vantagens do *MiddleWhere* são:

- Múltiplos sistemas de localização: suporta diversos tipos de sistemas de localização, tais como, GPS, Infra-Vermelho e IEEE 802.11;

- Confiança da informação: a qualidade da informação da localização depende de sua atualização. Conforme o tempo passa a qualidade da informação diminui. O *MiddleWhere* manipula a degradação da qualidade de localização reduzindo a confiança de localização conforme o tempo;

- Modelo de localização híbrido: permite que os dispositivos sejam localizados tanto por um modelo de localização física quanto por localização simbólica;

- Modelos de interação: aplicações sensíveis à localização podem interagir com o *MiddleWhere* através de duas formas. Podem perguntar por dispositivos que estão em alguma região e/ou receber a informação de quando um objeto acessa uma área (*triggers*);

- Localização por região ou por objeto: aplicações podem requisitar quais os objetos que estão em uma região ou perguntar aonde está um objeto em especial.

A integração do Gaia com o *MiddleWhere* é realizada através de um *Gaia Service*. As aplicações desenvolvidas em Gaia podem descobrir o serviço de localização consultando o *Gaia Space Repository*, que provê a lista dos serviços disponíveis. As aplicações podem invocar ações do serviço de localização diretamente. É possível também definir um nível de confidencialidade nas consultas, proporcionando maior segurança as aplicações. Na Figura 3.3 é apresentada uma visão da arquitetura do *MiddleWhere*.

3.3 one.world

O one.world [17] é uma plataforma desenvolvida pela Universidade de Washington. Destina-se a computação *pervasiva*, ou seja, à execução de aplicações em pequenos dispositivos móveis cooperando entre si na execução de tarefas. Este ambiente oferece um modelo de programação a ser seguido na construção destas aplicações, além de alguns serviços básicos comuns, com ênfase na mobilidade de código.

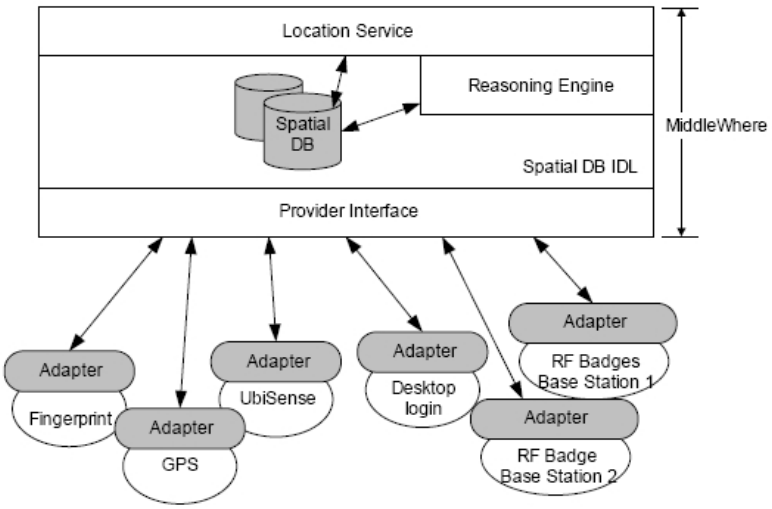

Figura 3.3 – Arquitetura do *MiddleWhere* [2]

No one.world o serviço de localização é denominado *Device Access Service* (DAS). O DAS coleta informações dos sensores (RFID) e converte para dados e eventos de aplicações one.world. Os eventos então são repassados ao *proximity service*, que realiza o *tracking* dos dispositivos.

3.4 PLACE

O PLACE *(Pervasive Location Aware Computing Environments)* [18, 19, 20] é um projeto desenvolvido pela Universidade Purdue. O foco está em prover um suporte a aplicações sensíveis à localização.

Seu objetivo é combinar funcionalidades de múltiplos sistemas de localização, como IEEE 802.11, GPS e aGPS. A principal funcionalidade da arquitetura PLACE é a habilidade de usuários móveis realizarem consultas a servidores de localização. A informação de localização é atualizada constantemente nesses servidores, sendo adquirida junto aos sistemas de localização. A tarefa de atualização da informação de localização pelos servidores é denominada "*continuous queries*".

O grande desafio técnico do PLACE consiste em providenciar serviços baseados em consultas aos servidores, mantendo escalabilidade, tempo de resposta e economia de recursos como largura de banda, memória e bateria dos dispositivos móveis. Dispositivos móveis no PLACE conhecem a sua localização. Os usuários do sistema possuem um perfil, onde podem incluir sua identidade e seus interesses. Na Figura

3.4 é apresentada uma tela do PLACE, utilizando o sistema de localização GPS.

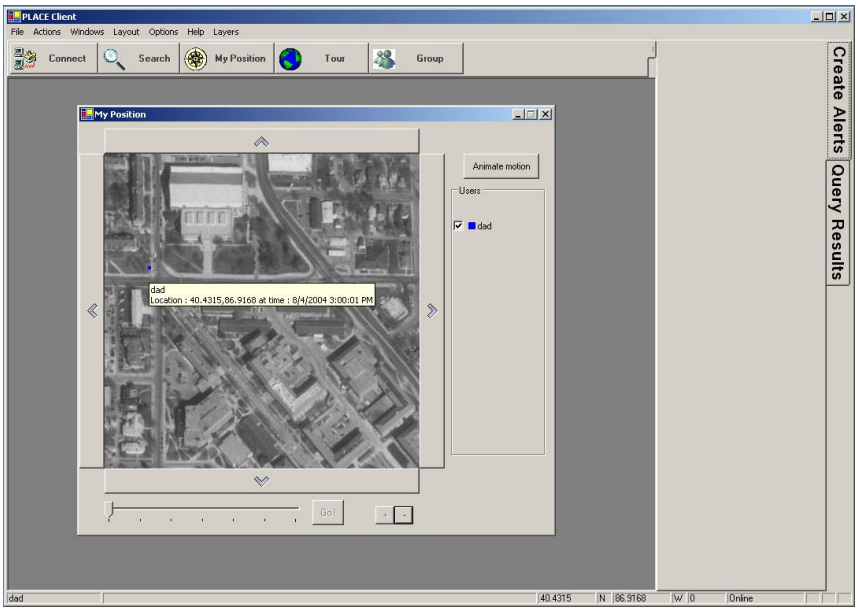

Figura 3.4 – Tela do sistema PLACE utilizando GPS

Atualmente o projeto conta com um protótipo em pequena escala, que utiliza o sistema de localização GPS. O servidor realiza o *tracking*, monitoração e responde a consultas de dispositivos no campus da Universidade Purdue.

3.5 Conclusões

Com o estudo feito neste capítulo foi possível comprovar que, com as devidas variações de estratégia, todos os projetos tentam solucionar as questões de posicionamento de dispositivos, com um interface voltada para diversas tecnologias de localização, o que permite uma melhor precisão das informações. Com exceção do projeto one.world, que tem como padrão a identificação dos dispositivos através da tecnologia RFID.

O projeto Aura dispõe do SIA, uma interface que realiza a busca de informações através de consultas semelhantes à SQL. Isso torna possível buscar informações do contexto como um todo. Se a consulta é por informações de posicionamento, então é repassada ao agente SNMP, que irá coletar essas informações junto a sensores/pontos de acesso. A limitação da utilização desta abordagem é que a requisição do agente SNMP ao dispositivo pode ser perdida,

porque a arquitetura SNMP utiliza o protocolo UDP para transporte, que não tem confiabilidade na entrega dos pacotes.

Entre os trabalhos pesquisados, o projeto Gaia possui o *MiddleWhere*, com a maior número de referências relacionadas com localização de dispositivos. Um recurso interessante é o grau de confiança da informação de localização, e é utilizado quando um dispositivo não é mais encontrado em nenhum sistema de localização. Quanto maior o tempo em que um equipamento não é encontrado, menor é o grau de confiança da informação. O grau de confiança é repassado como um campo adicional junto a resposta do *MiddleWhere*. Outro recurso que merece destaque é a modularização dos sistemas de localização, permitindo que sejam adicionados novas tecnologias sem alterar o protocolo.

O PLACE tem como objetivo a economia de recursos na aquisição de informações de posicionamento. Isso é relevante porque o consumo de bateria em dispositivos móveis é um requisito fundamental para o sucesso do projeto. No PLACE a aplicação é responsável pela busca da informação de posicionamento junto ao servidor de localização.

Capítulo 4

Holoparadigma

O Holoparadigma, de forma abreviada Holo, é um modelo multiparadigma orientado ao desenvolvimento de sistemas distribuídos e de computação móvel [6, 8]. Seus aspectos básicos já foram definidos, porém ainda há elementos a serem desenvolvidos e validados para a obtenção de uma plataforma completa para o desenvolvimento de sistemas ubíquos e móveis [3]. Os estudos relacionados com o Holo envolvem os seguintes tópicos de pesquisa: multiparadigma, sistemas *blackboard*, redes de computadores, sistemas distribuídos e computação móvel. Uma nova linguagem baseada no modelo permite a criação de programas usando os conceitos propostos. A Hololinguagem suporta concorrência, mobilidade, *blackboards* [54] hierárquicos e programação multiparadigma. Atualmente existem duas plataformas de execução: (1) os programas podem ser convertidos para Java usando uma ferramenta denominada HoloJava; (2) os programas podem ser compilados para um *bytecode* específico e executados usando uma máquina virtual criada no contexto do projeto. A seção 4.4 apresenta a taxonomia proposta para a classificação da unidade de modelagem do Holoparadigma.

4.1 Principais Conceitos

O Holoparadigma possui como unidade de modelagem o ente e como unidade de informação o símbolo. Um ente elementar (Figura 4.1a) é organizado em três partes: interface, comportamento e história. Um ente composto (Figura 4.1b) possui a mesma organização, no entanto, suporta a existência de outros entes na sua composição (entes componentes). Cada ente possui uma história. A história

fica encapsulada no ente e, no caso dos entes compostos, é compartilhada pelos entes componentes. Sendo assim, podem existir vários níveis de encapsulamento da história. Os entes acessam somente a história em um nível. A composição varia de acordo com a mobilidade dos entes em tempo de execução. A Figura 4.1c mostra um ente composto de três níveis e exemplifica a história encapsulada.

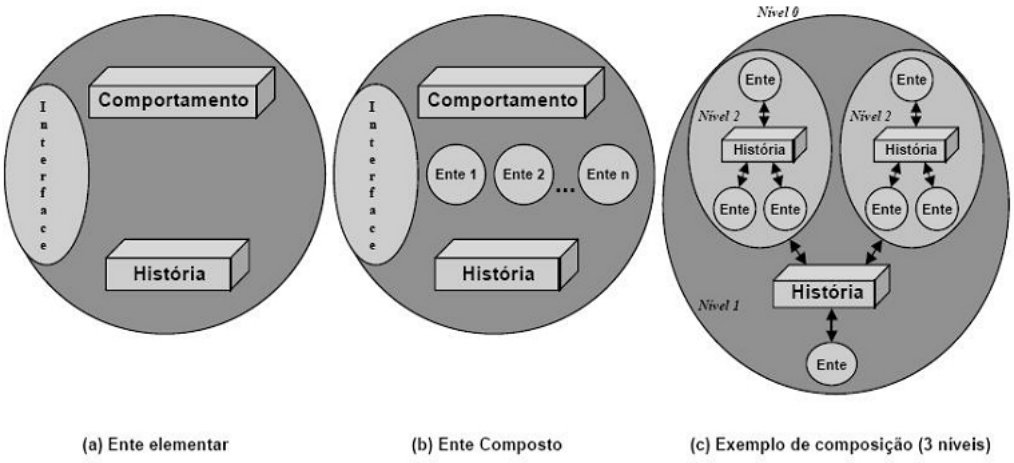

Figura 4.1 – Entes Elementar, Composto e Composição em 3 níveis

No escopo de sistemas distribuídos, um ente pode assumir dois estados de distribuição: centralizado ou distribuído. A Figura 4.2b mostra um possível estado de distribuição para o ente mostrado na Figura 4.1c. Neste caso, a história atua como uma memória compartilhada distribuída (DSM).

A mobilidade é a capacidade que permite o deslocamento de um ente. O Holoparadigma utiliza dois tipos de mobilidade (código e lógica). A mobilidade lógica relaciona-se com o deslocamento em nível de modelagem, ou seja, sem considerações sobre a plataforma de execução. Neste contexto, um ente se move quando cruza uma ou mais fronteiras de entes. A mobilidade de código relaciona-se com o deslocamento entre nodos de uma arquitetura distribuída. Neste contexto, um ente se move quando se desloca de um nodo para outro. A Figura 4.2a exemplifica uma possível mobilidade lógica no ente apresentado na Figura 4.1c. As mobilidades lógica e de código são independentes. A ocorrência de um tipo de deslocamento não implica a ocorrência do outro. Merece atenção o caso da mobilidade de código não implicar obrigatoriamente a mobilidade lógica. A Figura 4.2b mostra uma mobilidade de código sem mobilidade lógica.

A Hololinguagem integra os paradigmas imperativo, em lógica e orientado a

objetos. Ela busca não somente o suporte aos paradigmas integrados, mas também as novas oportunidades que surgem através da integração. A linguagem utiliza representação simbólica, unificação e não possui tipos. Estas características são herdadas do paradigma em lógica. A linguagem suporta programação de alta ordem, ou seja, símbolos representando entes e suas partes (interface, ações, componentes e história) podem ser manipulados através de variáveis e operações da linguagem. Entes elementares, compostos e vários níveis de *blackboards* podem ser utilizados.

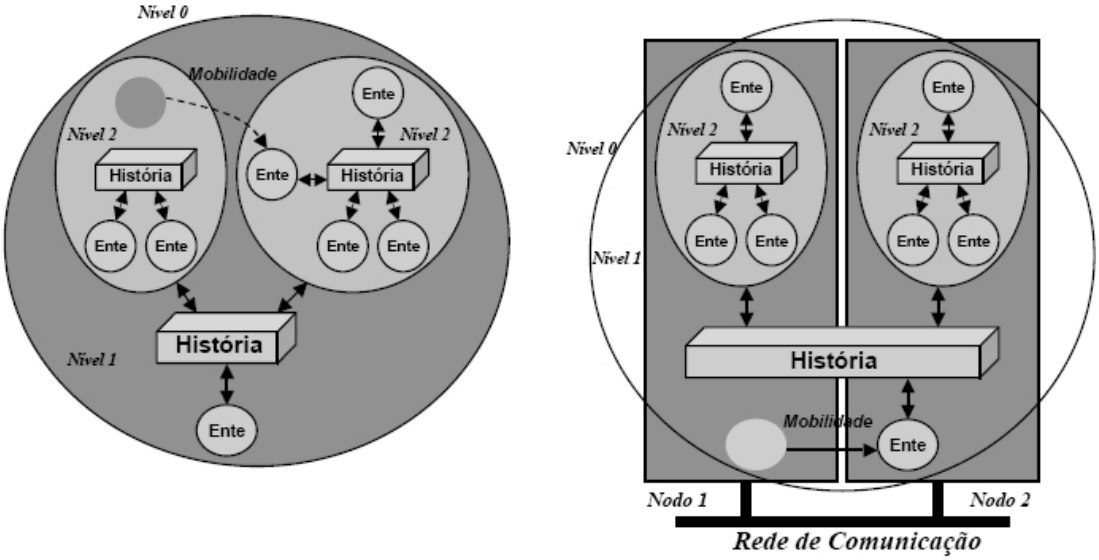

Figura 4.2 – Exemplo de mobilidade no Holoparadigma

4.2 Holo Tree

A execução de um programa cria uma estrutura hierárquica de entes, denominada Árvore de Entes (*HoloTree*) [6, 8]. A *HoloTree* implementa o encapsulamento de entes em níveis de composição, conforme proposto pelo Holoparadigma. Além disso, a *HoloTree* suporta o aspecto dinâmico da política de grupos, mudando continuamente durante a execução de um programa.

A Figura 4.3a exemplifica a *HoloTree* para a organização em níveis mostrada na Figura 4.1c. Um ente composto possui ligações com seus entes componentes, os

quais ficam localizados no nível abaixo. Os entes componentes possuem acesso a história e ao comportamento do composto no qual estão inseridos. Por sua vez, o composto possui acesso aos comportamentos dos seus componentes. Além disso, um ente possui acesso ao comportamento dos demais entes no mesmo nível.

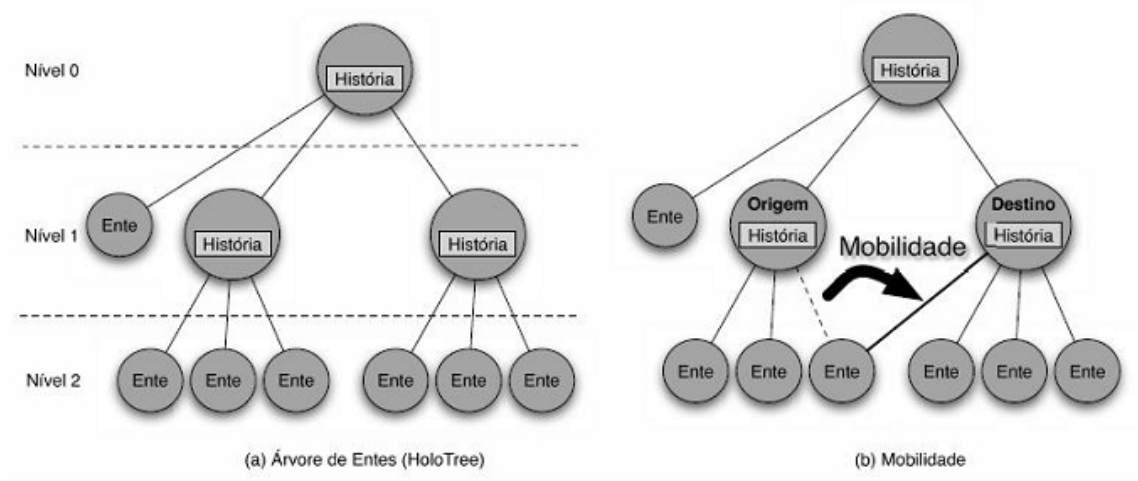

Figura 4.3 – Organização hierarquica no Holoparadigma - *HoloTree*

Quando a mobilidade ocorre, torna-se necessária a mudança da visão do grupo dos entes envolvidos. O ente movido possui uma nova visão (novos entes no mesmo nível e um novo composto acima dele). Se o movido for um ente composto, a visão dos seus componentes não muda. A mobilidade implica a atualização da composição dos entes origem e destino. Além disso, os componentes de um ente podem ser entes compostos (grupos de grupos). A mobilidade de um ente elementar equivale a realocação de uma folha da árvore e a mobilidade de um composto transfere um ramo contendo vários entes. A Figura 4.3b apresenta a mudança que ocorreria na *HoloTree* para a mobilidade na Figura 4.2a.

4.3 HNS e HVM

Atualmente o suporte de execução à Hololinguagem é fornecido por dois sistemas: A *Holo Virtual Machine* (HVM) [26] e o *Holo Naming System* (HNS). A HVM é uma máquina virtual para executar programas em Holo, e possui um conjunto de instruções que foram criadas especificamente para dar suporte as funcionalidades do Holoparadigma. O HNS foi criado para possibilitar que várias

HVMs, dispostas em máquinas diferentes, sejam capazes de interagir e executar uma mesma aplicação. Este sistema permite a implementação de sistemas de computação distribuída. O HNS controla a execução distribuída de entes, fornecendo informações necessárias para as HVMs, e permitindo a comunicação entre os mesmos.

A *HoloTree* existente nas HVMs de cada dispositivo é apenas uma visão parcial do cenário, no qual apenas a HNS irá conhecer a estrutura completa da árvore, já que todas as HVMs reportam ao sistema HNS quando realizam alguma alteração em suas *HoloTrees*. Esta estrutura mantida pelo HNS é denominada *Distributed HoloTree* (DHoloTree).

O servidor deve manter-se sempre atualizado, com relação à execução da aplicação, para isto, algumas instruções ao serem interpretadas pela HVM, geram mensagens para o HNS. Quando a HVM executa a instrução *clone* (criação de um novo ente), o HNS é informado sobre o novo ente e o contexto na qual ele se encontra. Após ser criado, o ente pode ser movido entre os contextos da aplicação. A mobilidade é fornecida na linguagem através da instrução *move*. Esta instrução é responsável pela dinamicidade das aplicações Holo, e seu monitoramento é fundamental para que o HNS esteja sincronizado.

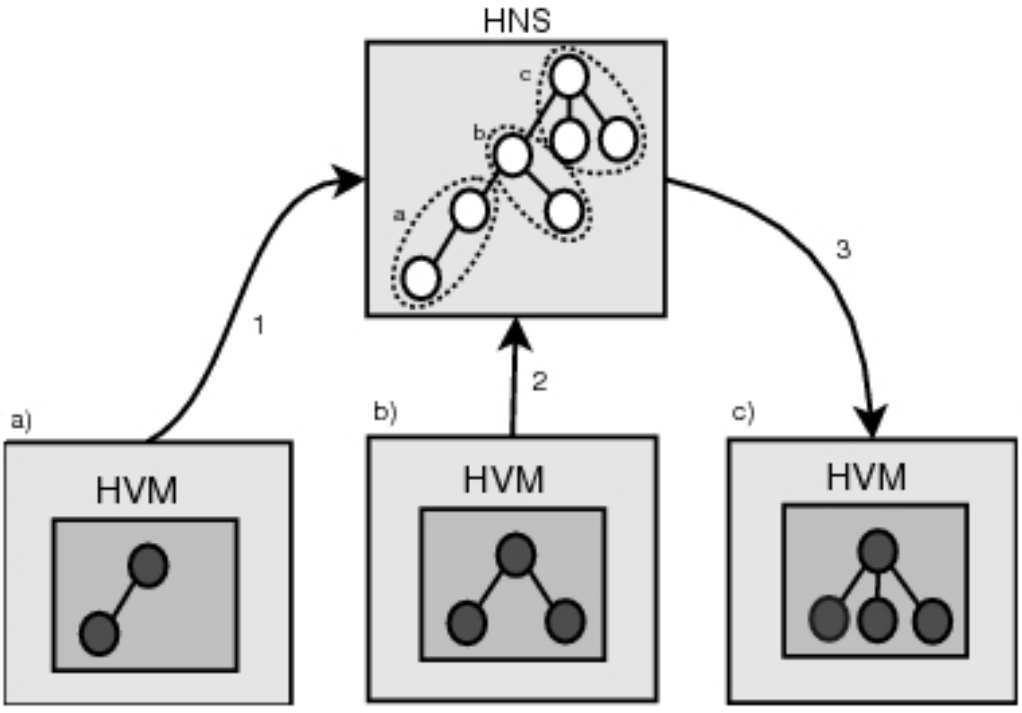

Figura 4.4 – Figura apresentando funcionamento do HNS [3]

Na Figura 4.4 é apresentado um ambiente de execução utilizando o HNS e as possíveis interações com uma ou mais HVMs. No *passo 1* a HVM *a* informa a criação de um ente para o HNS. Tendo feito isso, para que dois entes possam se comunicar basta que eles tenham acesso à história de um ente em comum. Os *passos 2* e *3* exemplificam as HVMs *b* e *c*, interagindo com o servidor para acessar a história de um ente.

4.4 Taxonomia de entes para mobilidade

Um ente pode representar qualquer entidade do mundo físico, por exemplo um prédio ou uma sala. Um ente também pode ser virtual, por exemplo um programa. Com este alto nível de abstração torna-se necessária a criação de uma taxonomia que leve em conta o seu tipo (físico ou virtual) e o seu estado de mobilidade (estático ou dinâmico). Esta taxonomia é necessária para realizar as alterações no ambiente de execução Holo, possibilitando a integração com aplicações sensíveis à localização. Neste sentido, propõe-se a seguinte taxonomia de entes:

- *Virtual Beings (VB):* é todo e qualquer ente que é abstrato, ou seja, que não possui representação no mundo físico. Os VBs são subdivididos em duas categorias:
  - *Virtual Static Being (VSB):* é todo ente virtual que está fixo a um determinado contexto. Os VSBs podem trocar de contexto, mas isso não é um evento que ocorre freqüentemente. Um exemplo de ente VSB pode ser uma agenda pessoal ou uma lista de contatos. Na Figura 4.5a é apresentada a sua representação gráfica;
  - *Virtual Dynamic Being (VDB):* é todo ente virtual que troca de contexto freqüentemente durante a execução de um programa Holo. Seu tempo de permanência em um contexto é relativamente curto, pois quando termina de executar uma ação parte para outro contexto. Um ente para coleta de informações sobre o consumo de água e/ou energia elétrica é um exemplo de ente VDB. Sua representação gráfica é apresentada na Figura 4.5b;
- *Physical Beings (PB):* é todo e qualquer ente que está presente no contexto

Holo, representando uma estrutura ou objeto do mundo real. São subdividos em duas categorias:

- *Physical Static Being (PSB):* é todo ente físico que não possui uma mobilidade freqüente[1]. Pode-se definir como exemplo de PSBs um computador *desktop*, uma sala, um prédio ou qualquer coisa que é representada no mundo real, mas que tem sua posição fixa em toda ou maior parte do tempo. Sua representação gráfica pode ser visualizada na Figura 4.5c;
- *Physical Dynamic Being (PDB):* é todo ente físico que possui uma mobilidade freqüente, que troca diversas vezes de contexto durante a execução de um programa Holo. Como exemplos de PDBs pode-se citar *laptops*, *handhelds*, PDAs ou *tablet pcs*. Na Figura 4.5d é apresentada a sua representação gráfica.

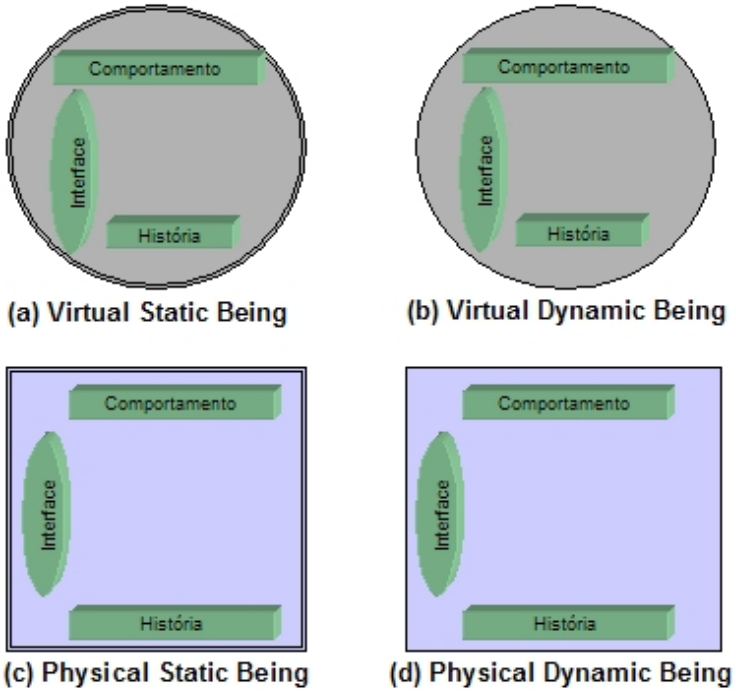

Figura 4.5 – Tipos de Entes no Holoparadigma

[1]Mobilidade freqüente: mobilidade que representa o possível deslocamento de um dispositivo como *notebook*, PDA, *tablet pc* ou celular durante sua utilização

Um exemplo para melhor comprensão da taxonomia proposta é a aplicação denominada *"MobiLab Pervasive Learning"*. O seu objetivo é relacionar conteúdo didático conforme a localização de um equipamento. Cada ente que representa um equipamento possui o seu perfil armazenado na sua história. Neste perfil estão armazenados dados de identificação pessoais, e uma lista com seus tópicos de interesse. Existe no sistema o ente "*Engine*". O *Engine* é do tipo VDB, sendo responsável pelo processamento de perfis. Quando o *Engine* acessa um contexto e encontra assuntos de interesse mútuo em dois perfis diferentes, é apresentado uma mensagem para os usuários, mostrando que eles tem interesse pelo mesmo assunto.

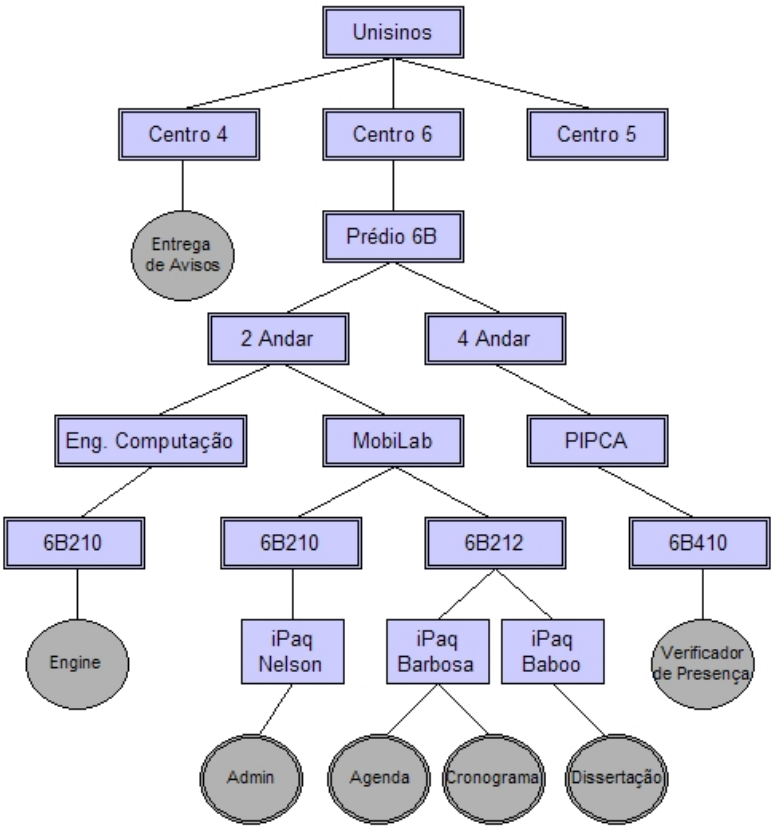

Figura 4.6 – Exemplo da *HoloTree* para a aplicação "*MobiLab Pervasive Learning*"

Na Figura 4.6 é apresentado um exemplo da *HoloTree* para esta aplicação. É possível observar os ambientes deste sistema (entes PSB), organizados de forma hierárquica. Os entes PDBs "*iPaq Nelson*", "*iPaq Barbosa*" e "*iPaq Baboo*" representam os dispositivos móveis de usuários deste sistema, e possuem mobilidade freqüente.

Os entes virtuais também são apresentados na Figura 4.6. Os entes que realizam tarefas de verificação e percorrem toda ou parcialmente a *HoloTree* (entes VDB) são: "*Engine*", "*Entrega de Avisos*" e "*Verificador de Presença*". O ente "*Engine*" representa o analisador de perfil de usuários, conforme a sua localização. "*Entrega de Avisos*" é o ente encarregado da tarefa de avisar os entes que estão em certos contextos sobre eventos que acontecem no ambiente. O ente "*Verificador de Presença*" valida a presença física de cada aluno, permitindo assim um controle automático desta tarefa. Os entes VSB "*Admin*", "*Agenda*", "*Cronograma*" e "*Dissertação*" são aplicações que estão atreladas aos usuários, de domínio pessoal e somente mudarão de contexto se forem informadas pelo seu usuário.

4.5 Conclusões

Nesta capítulo foram apresentados o Holoparadigma, e as tecnologias que criam o suporte à execução para este paradigma. Com o conhecimento adquirido, será possível projetar a integração de aplicações sensíveis à localização em Holo, permitindo a utilização das vantagens proposta pelo paradigma.

Foi proposta uma taxonomia para os tipos de entes presentes em ambientes sensíveis à localização, perante o seu perfil de mobilidade. No estudo de caso é apresentada uma aplicação de exemplo utilizando os entes propostos na taxonomia.

Atualmente o ambiente de execução do Holoparadigma não tem o suporte a aplicações sensíveis à localização. No próximo capítulo, será apresentado a proposta para a criação de um suporte a aplicações sensíveis à localização, utilizando o Holoparadigma e o seu ambiente de execução.

Capítulo 5

Modelo HLS

A necessidade de aprimorar o ambiente Holo para a execução de aplicações ubíquas, na qual um dos requisitos básicos é a consciência de contexto, é o foco deste trabalho. Neste capítulo é apresentado o modelo *Holo Locating System*, de forma abreviada HLS, que integra o suporte a aplicações sensíveis à localização no ambiente Holo, explorando os princípios do Holoparadigma.

5.1 Arquitetura da Solução

O HLS é um modelo, que integrado ao ambiente de execução Holo, permite a criação de aplicações sensíveis à localização. O HLS possui três componentes: o que oferece o suporte à localização de dispositivos é o Servidor de Localização, o responsável pela atualização da aplicação, chamado de *Context Changer* e uma ferramenta para administração de localização de equipamentos por árvore de contextos, a *Holo Tree View*.

Na Figura 5.1 é apresentada a estrutura de requisições do modelo. Pode-se observar a comunicação dos dispositivos móveis com o Servidor de Localização, que é o responsável por manipular e gerenciar informações oriundas de sistemas de localização. Na Figura 5.1 o Servidor de Localização recebe dos equipamentos móveis a informação de potência de sinal das antenas de pontos de acessos conhecidos. O servidor também recebe as informações dos computadores *desktop*, através do protocolo SNMP. Baseada nessa informação é realizado um cálculo para saber em que local o dispositivo encontra-se no momento. O *Context Changer* é integrado ao HNS, que é o software que mantém a *Holo Tree* distribuída. Quando ocorrem

mudanças no posicionamento dos dispositivos, o HNS recebe uma mensagem do Servidor de Localização, informando qual o novo contexto do dispositivo. O *Context Changer* também pode realizar consultas ao Servidor de Localização, para economizar recursos de rede e processador. Finalmente a *Holo Tree View* (HTV) recebe informações no formato de árvores de contextos, as quais podem ser oriundas do Servidor de Localização e do HNS (*HoloTree*).

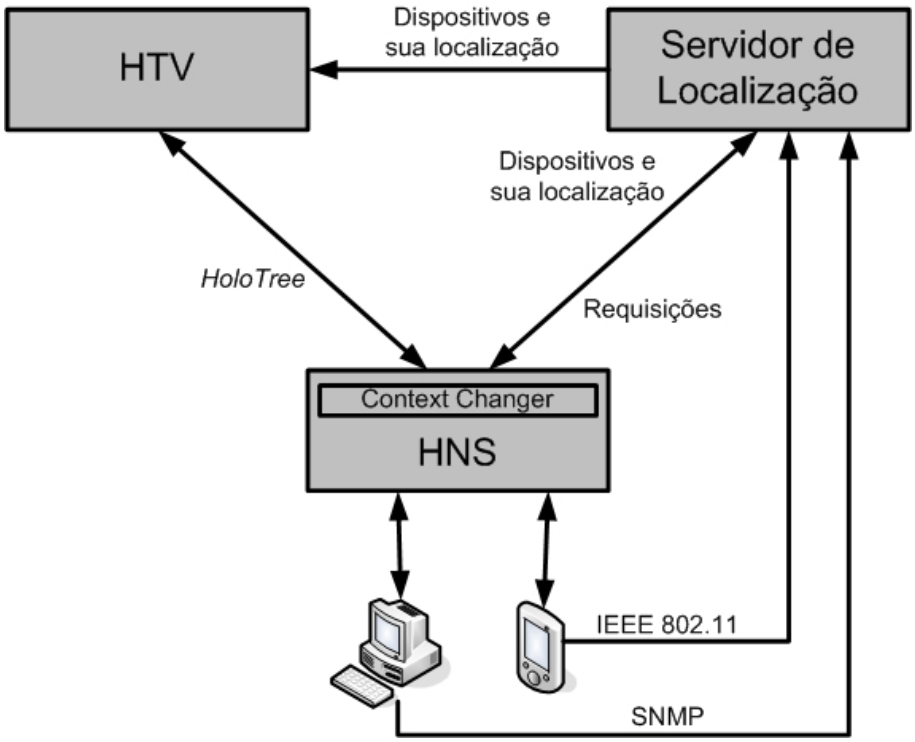

Figura 5.1 – Arquitetura do *Holo Locating System*

5.2 Servidor de Localização

O Servidor de Localização, denominado SELIC, é um servidor de localização que permite que aplicações acessem informações sobre a localização de dispositivos. A sua tarefa é integrar diversos sistemas de localização (GPS, RFID, IEEE 802.11, etc) e disponibilizar as informações através de um protocolo padrão. Cada sistema de localização é um módulo no SELIC. Esses módulos são adicionados ao SELIC através da implementação de uma classe abstrata. As informações de localização dos dispositivos são disponibilizadas através de uma interface de alto nível.

A API do SELIC foi concretizada através de uma interface de *Web Services*.

Esta escolha foi feita pela facilidade de integração com outras aplicações. *Web services* são uma solução utilizada na integração de sistemas e na comunicação entre aplicações diferentes. Com esta tecnologia é possível que novas aplicações possam interagir com aquelas que já existem e que sistemas desenvolvidos em plataformas diferentes sejam compatibilizadas. As bases para a construção de um *Web Service* são os padrões XML e SOAP. Os dados são transferidos no formato XML, encapsulados pelo protocolo SOAP. O transporte dos dados é realizado, normalmente, via protocolo HTTP, mas o padrão não determina o protocolo de transporte.

A precisão das informações de posicionamento é uma questão importante, e para isso, o SELIC tem suporte a múltiplos sistemas de localização, o que torna possível comparar as informações e escolher a de maior coerência. Na Figura 5.2 é apresentado uma situação em que o SELIC possui como módulos os sistemas de localização GPS, IEEE 802.11 e RFID.

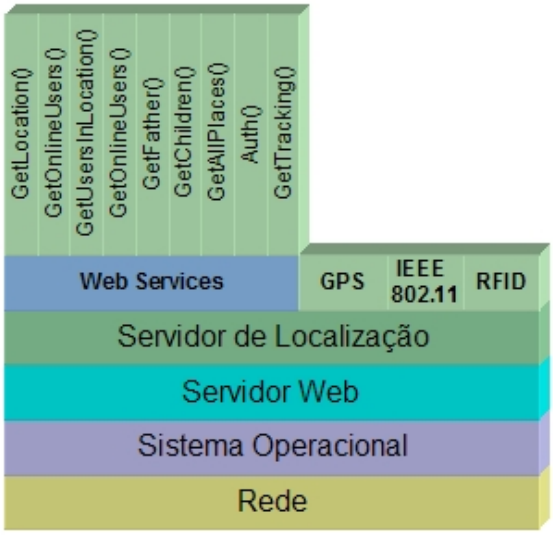

Figura 5.2 – Arquitetura do SELIC

Com base no exemplo da Figura 5.2, pode-se descrever como é realizado o cálculo de posicionamento para os seguintes sistemas de localização:

- GPS: É repassada a coordenada obtida junto ao *receiver*. Esta coordenada é processada no SELIC seguindo um mapeamento cadastrado junto a um banco de dados;

- IEEE 802.11: São repassadas as informações do nível de sinal dos pontos de acesso que o sistema tem disponibilidade. O cálculo do posicionamento é feito com base num banco de dados que contém o mapeamento. No mapeamento são colhidas amostras da potência de sinal de pontos de acesso conhecidos para cada contexto. O dispositivo precisa ter o sinal de no mínimo três pontos de acesso para obter informações de posicionamento;

- RFID: Cada leitor é responsável por identificar dispositivos em uma sala. O leitor é consultado em uma faixa de tempo, para obter os dispositivos que estão sendo identificados no seu espaço.

O SELIC tem suporte a *tracking* de dispositivos. Quando o *tracking* estiver habilitado para um equipamento, as informações de posicionamento são registradas em uma base de dados para futuras consultas de aplicações. Um exemplo de utilização de *tracking* pode ser a auditoria para descobrir o perfil de deslocamento de um equipamento. O tempo de *tracking* de um dispositivo é mantido com base em informações armazenadas em um arquivo XML. Nesse arquivo é possível especificar o tempo em que a informação será armazenada. Por exemplo, se o *tracking* do dispositivo *"iPaq 3"* é de 4 meses, as informações acima deste tempo serão descartadas. Com isso é possível manter a base de dados com informações relevantes, melhorando o desempenho em consultas e economizando recursos de armazenamento e memória.

A interface do SELIC é disponibilizada através dos seguintes métodos:

- *Auth*: Método para realizar a autenticação no sistema. O cliente do SELIC envia o nome de usuário e senha ao servidor, permitindo a utilização do serviço de localização;

- *GetAllPlaces*: Método para retornar a lista de locais que podem usufruir do sistema de localização;

- *GetChildren*: Método para retornar os contextos que são filhos do local passado como parâmetro;

- *GetFather*: Método para receber o local que está no topo da árvore de localização;

- *GetLocationByUserId*: Retorna a localização atual pelo *User Id* do usuário;
- *GetLocationByUserName*: Retorna a localização atual pelo nome do usuário;
- *GetOnlineUsers*: Informa os usuários que estão *online* no sistema;
- *GetUsersInLocation*: Retorna uma lista com todos os usuários que estão no local passado como parâmetro;
- *GetTracking*: Método para receber o histório de deslocamento de um dispositivo.

5.3 *Context Changer*

A primeira medida para construir uma aplicação sensível a localização no Holoparagima é a construção da *HoloTree*. A *HoloTree* deve ser baseada na árvore de contexto dos ambientes do SELIC. Isso é necessário para que o programa holo seja um mapeamento exato da situação do SELIC, mantendo a integridade referencial da aplicação com os locais do mundo físico. Os serviços de cada ambiente devem ser mapeados como ações para o seu representante na estrutura, permitindo que sejam agrupados e organizados no mesmo contexto.

O *Context Changer*, de forma abreviada CC, é um módulo integrado ao HNS com função de manter os entes da *HoloTree* que representem dispositivos móveis sincronizados com sua localização física. Os entes que representam equipamentos (PDBs) tem seu pai alterado conforme mudam de contextos no mundo físico. Quando um dispositivo troca o seu local no ambiente físico, esse ente é realocado na *HoloTree*, permitindo que o mesmo acesse o seu pai para utilizar os recursos disponíveis no contexto atual. Na Figura 5.3 é apresentado o funcionamento do *Context Changer*. Nela o *parser* processa as requisições que chegam no HNS. Quando o *parser* identifica que uma mensagem é enviada pelo SELIC, ele realiza uma chamada para o módulo *Context Changer* passando como parâmetro a mensagem. A mensagem então é processada pelo CC, que altera somente os entes que representam equipamentos no SELIC.

O CC é quem mantém sincronizado os entes que estão nas HVMs e que representem dispositivos físicos no mundo real. A base do mapeamento é um arquivo

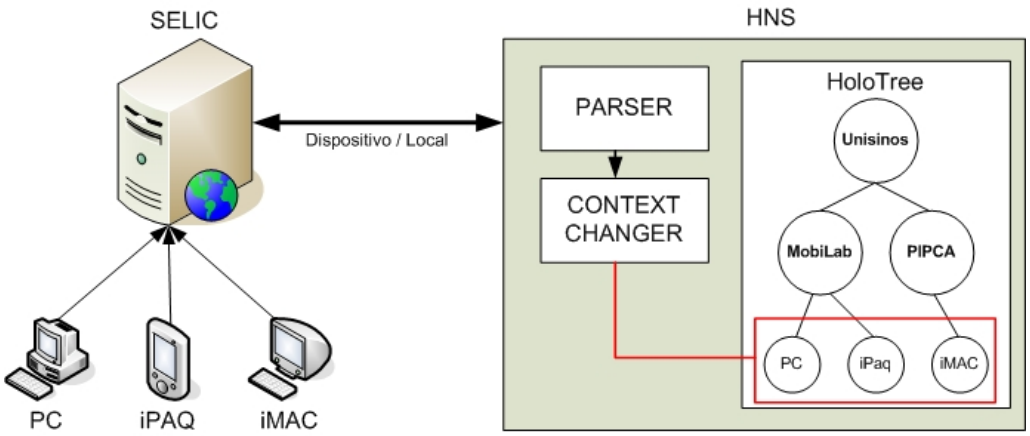

Figura 5.3 – Apresentação do módulo *Context Changer*

em XML que descreve quais são os nomes dos entes que representam equipamentos. A utilização do arquivo de mapeamento torna fácil a integração de novos dispositivos ao sistema, permitindo adicionar ou remover equipamentos conforme a necessidade da aplicação. Na Figura 5.4 é apresentado um *snapshot* do momento inicial de uma aplicação Holo e dos dispositivos encontrados pelo SELIC.

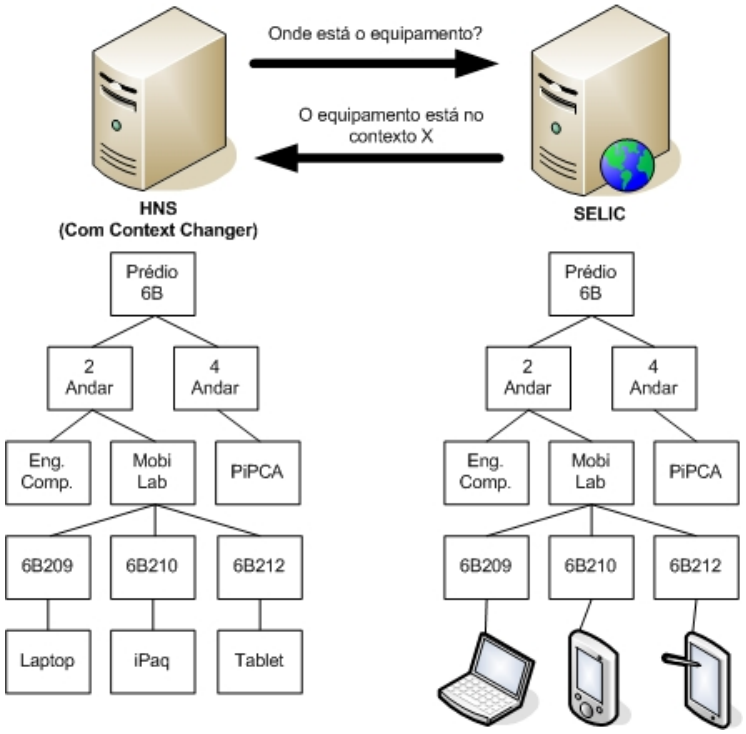

Figura 5.4 – Exemplo da situação inicial de uma aplicação Holo e o SELIC

Quando o SELIC detecta que um dispositivo teve seu contexto alterado, ele repassa esta informação ao HNS. Esta informação é processada pelo *Context Changer*

que é encarregado de pesquisar o ente e alterar o seu pai na *DHoloTree*, gerando uma mobilidade na árvore. Depois de efetuar esta operação, se o novo pai possuir ações relevantes ao contexto, elas já poderão ser acessadas pelo ente filho que foi movido. Na Figura 5.5 é apresentado o novo estado da aplicação e do SELIC após acontecer a mobilidade de alguns dos dispositivos apresentados na Figura 5.4.

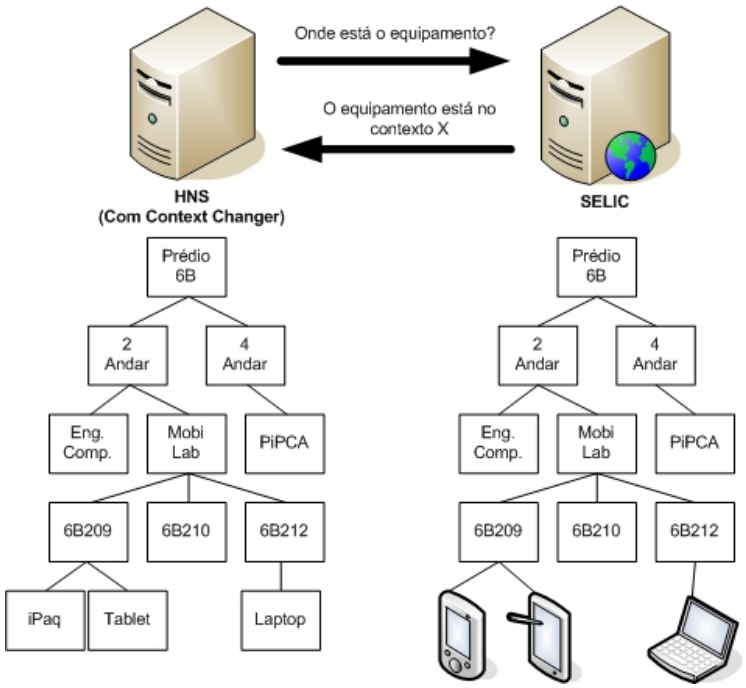

Figura 5.5 – Exemplo da de uma mobilidade sobre o exemplo da Figura 5.4

Uma alternativa para o recebimento de informações do SELIC, é quando o CC pode consultar sua API em um determinado período de tempo. Nesta abordagem o *Context Changer* realiza requisições ao SELIC pedindo a localização dos equipamentos. Essas requisições são realizadas em um intervalo de tempo configurado pelo usuário. Essa solução é ideal para ambientes onde existe uma grande quantidade de equipamentos, principalmente pela economia de recursos e escalabilidade do sistema.

5.4 *Holo Tree View*

A existência de um servidor de localização traz novas funcionalidades que podem ser exploradas para a administração de usuários e dispositivos. Um fator que necessita especial atenção e pode ser explorado em ambientes ubíquos é a

administração de recursos e equipamentos. Após verificar a necessidade de uma ferramenta para visualização da *HoloTree*, que está presente no HNS, e da árvore de ambientes, disponível no SELIC, foi desenvolvida a *Holo Tree View*, de forma abreviada HTV, que é capaz de apresentar as informações baseadas em árvores de contexto de forma visual. A exibição da árvore mantém a mesma formatação apresentada na taxonomia de entes proposta no capítulo anterior.

Com a *Holo Tree View* é possível acompanhar o deslocamento dos equipamentos ou entes. Com essa informação o administrador pode verificar algum equipamento que está em local não apropriado, identificar locais que estão superlotados ou acompanhar o caminho que um ente dinâmico está percorrendo na *HoloTree*.

Outra funcionalidade que pode ser utilizada é o detalhamento do *tracking* de um dispositivo. Quando a HTV utiliza o *tracking* de um dispositivo como fonte de informação, o percurso é apresentado com uma seqüência numérica. Essa seqüência também trás o tempo que o dispositivo permaneceu no local.

A HTV também é integrada ao HNS. Quando utilizada deste modo, ela realiza uma chamada ao sistema requisitando o estado atual da *DHoloTree*. Ao receber a informação a *Holo Tree View* é encarregada do processamento e apresentação visual. A partir desta integração o administrador pode encontrar máquinas que estejam sobrecarregadas, com muitos entes sendo executados na mesma HVM.

5.5 Conclusões

Neste capítulo foi apresentado o modelo para suporte a aplicações sensíveis à localização chamado *Holo Locating System*, que é integrado ao ambiente de execução do Holoparadigma. A criação deste modelo permite a criação de aplicações sensíveis à localização no Holoparadigma.

O SELIC destaca-se por ter uma modularização dos seus sistemas de localização, o que permite adicionar novos sistemas sem a necessidade de alterar o protocolo de comunicação com as aplicações. A utilização de uma interface de comunicação baseada em *Web Services* permite também integrar o sistema a diferentes aplicações, sem a restrição de ser unicamente utilizada no Holoparadigma.

O *Context Changer* é o responsável pela sincronização das informações

providas pelo SELIC nas aplicações Holo. Nesta abordagem o próprio ambiente de execução atualiza as informações, focando o desenvolvedor no modelo da aplicação, e não em como ela será processada. Isso é possível porque a organização de um programa Holo é realizado através de árvores de contextos.

A *Holo Tree View* permite que os administradores acompanhem uma árvore de contexto durante a execução do SELIC ou de um programa Holo, que esteja sendo executado no ambiente distribuído (HNS).

No próximo capítulo serão abordadas questões de implementação do SELIC, *Context Changer* e *Holo Tree View*.

Capítulo 6

Aspectos de Implementação

Neste capítulo serão abordados detalhes da implementação do modelo HLS, proposto no capítulo anterior. O objetivo deste capítulo é detalhar informações e decisões de projeto para a construção do HLS.

6.1 SELIC

O servidor de localização foi implementado utilizando como ambiente de desenvolvimento o Microsoft Visual Studio 2005, utilizando a linguagem de programação C#. Essa escolha foi determinada pela utilização do Microsoft *.NET Framework* e a possibilidade de portabilidade para outras plataformas, como por exemplo Linux utilizando o compilador e *framework* Mono. Outro fator importante foi a integração do Microsoft Visual Studio 2005 com *Web Services*, possibilitando que o SELIC fosse utilizado com diferentes aplicações.

6.1.1 Arquitetura do SELIC

A arquitetura do SELIC é dividida em três partes:

- *Web Services*: Cada equipamento, sensor ou aplicação utiliza a interface de serviços baseada em *Web Services* para se comunicar com o SELIC. Nos *Web Services* estão os métodos para autenticação no sistema, consulta de posicionamento de dispositivos, consulta na árvore de ambientes entre outros. Na Figura 6.1 é apresentado a interface de comunicação em um navegador;
- Cliente: É o responsável pela autenticação no sistema e integração com os *Web*

Services. Após o usuário instalar o sistema no seu dispositivo móvel, é possível saber qual o local que o dispositivo encontra-se atualmente. Por exemplo, se o sistema de localização for o IEEE 802.11, é informada a potência de sinal das antenas de pontos de acesso conhecidos para os *Web Services*. Atualmente o cliente pode ser executado nos ambientes Windows (*PCs*, *Tablets*) e *Windows Mobile* (iPaqs) . Na Figura 6.2 é apresentado a tela de autenticação do sistema;

- *Engine*: É a principal parte do sistema. É responsável por realizar os cálculos de posicionamento dos equipamentos. Cada equipamento reporta informações dos sistema de localização ao *Web Service* do SELIC. Essa requisição é colocada numa fila, que então é processada pelo *Engine*. A escolha pela utilização do *Engine* é devido ao fato dos *Web Services* não guardarem o estado da conexão (*stateless*), impossibilitando a conexão assíncrona com outras aplicações. No *Engine* também pode ser definida a ordem de procedência dos sistemas de localização (caso tenha mais de um sistema sendo utilizado no momento). Isso permite ter maior controle sobre a localização dos dispositivos baseados na confiança da informação dos sistemas de localização. Os sistemas são acoplados ao *Engine* através da implementação de uma classe abstrata. Sendo assim, é possível adicionar novos módulos ao sistema sem a necessidade de alterações no sistema.

  Atualmente os sistemas de localização implementados são o IEEE 802.11 e o SNMP:

  - IEEE 802.11: Cada ambiente possui um mapeamento com a potência de sinal de antenas de pontos de acessos conhecidos. Para cada metro quadrado do ambiente são colhidos n pontos. Quanto maior o número de pontos, maior a precisão do sistema. Cada ponto contém informações dos pontos de acesso disponíveis e sua potência de sinal. O mapeamento é armazenado em um banco de dados. Quando o *Engine* processa uma requisição IEEE 802.11, ele busca no banco de dados do mapeamento, procurando por pontos que enquadram-se na potência de sinal de pontos de acesso da requisição. O ambiente que estiver com mais pontos, é a atual posição do dispositivo. No *Engine* também foi implementado uma análise estatística, para contornar o problema da constante variação de

potência de sinal. Essa análise leva em conta os últimos dez ambientes que o dispositivo foi encontrado, e é retornado o ambiente que o equipamento foi encontrado mais vezes. Essa análise busca apenas a informação dos últimos cinco minutos, podendo este tempo ser configurado pelo usuário;

- SNMP: O SNMP é utilizado no SELIC em equipamentos com pouca mobilidade, por exemplo computadores *desktop*. Cada equipamento é incluído numa tabela com o seu local atual, seu identificador de dispositivo, IP e senha. O *Engine* busca em determinado intervalo de tempo por todos dispositivos nesta tabela e realiza uma requisição SNMP *GET upTime* para ver se o dispositivo está online. Caso haja resposta, o dispositivo é registrado no *tracking* do sistema.

Na Figura 6.3 é apresentado o cálculo de posicionamento para um equipamento pelo *Engine*.

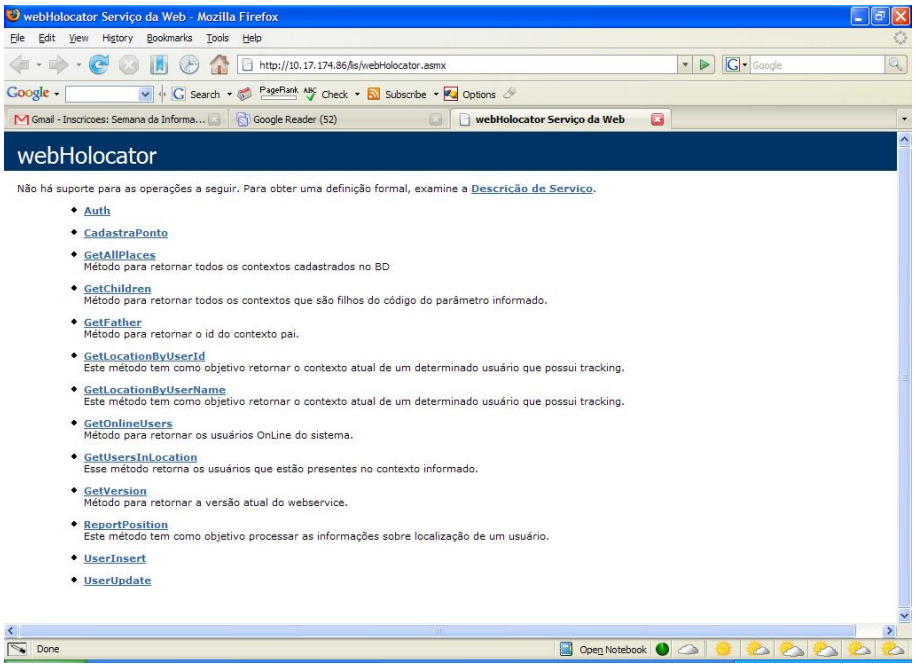

Figura 6.1 – Apresentação dos *Web Services* do SELIC

6.1.2 Precisão

Um dos fatores mais importantes de um servidor de localização é a sua precisão. Com base na precisão é possível criar o nível de detalhamento para as aplicações, como por exemplo, "*room level*", "*floor level*", "*build level*" ou "*area level*".

Figura 6.2 – Tela de autenticação no sistema SELIC (cliente)

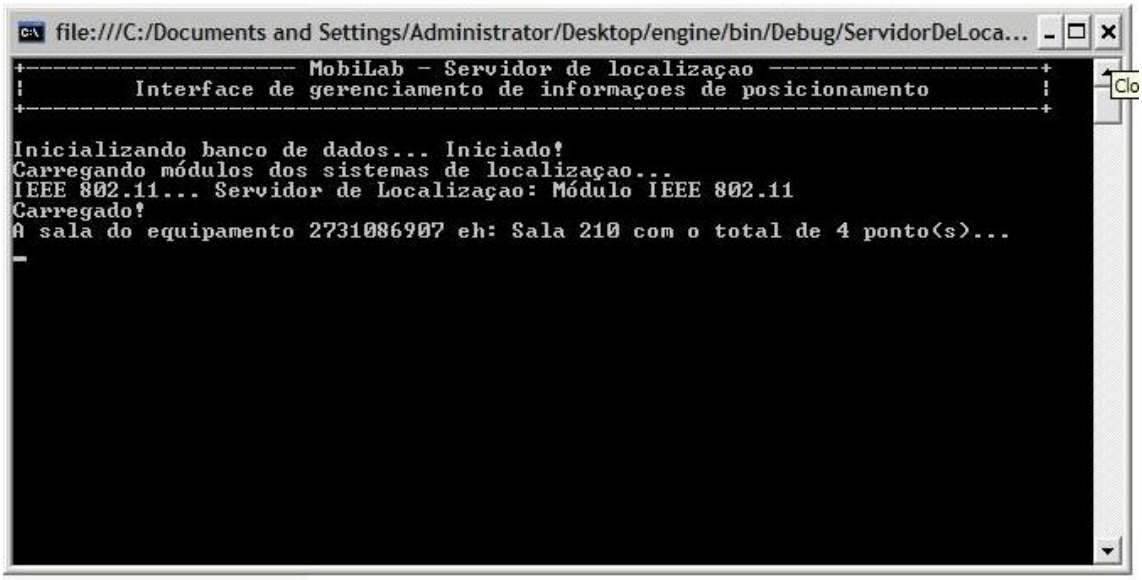

Figura 6.3 – *Engine* do SELIC realizando um cálculo de posicionamento

O SELIC foi criado para ter precisão "*room level*" utilizando como principal sistema de localização o IEEE 802.11. Para o cenário de testes, foi utilizado o segundo andar do prédio 6B da Unisinos, onde estão o curso de Engenharia da Computação e o MobiLab. A atual estrutura do andar possui quatro pontos de acesso, distribuídos conforme o mapa apresentado na Figura 6.4.

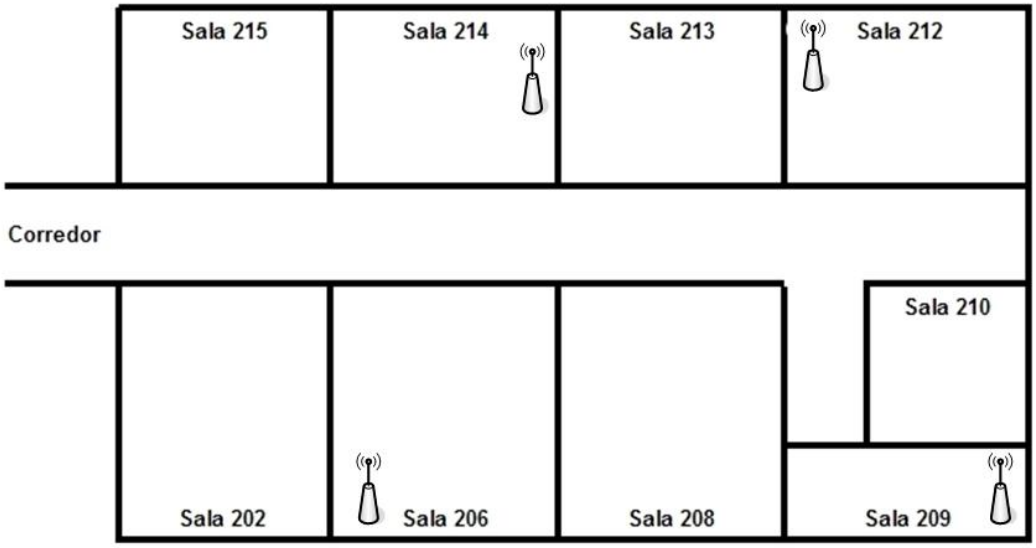

Figura 6.4 – Distribuição dos pontos de acesso no segundo andar do prédio 6B da Unisinos

Atualmente o SELIC possui precisão "*room level*". Isso é alcançado com base no mapeamento realizado no andar, onde foram captadas as potências de sinal das antenas para cada metro quadrado de cada sala. Quando o SELIC processa uma requisição do sistema de localização IEEE 802.11, ele verifica em qual local mapeado existe a maior quantidade de pontos encontrados conforme informações de sinal de potência que está na requisição. Esta localização então é armazenada temporariamente em um banco de dados.

A Figura 6.5 apresenta o gráfico de precisão do SELIC. Neste teste foram utilizadas três antenas como base de cálculo do posicionamento, ou seja, o cálculo era realizado somente quando o dispositivo tinha a potência de sinal de três pontos de acesso. O equipamento utilizado foi um *HP Compaq tc1100 Tablet PC*, com processador Centrino de 1.8 MHz, 512 MB de memória RAM e placa *wireless*

ORiNOCO SILVER PCMCIA. Foram escolhidos três locais para amostra na sala 6B208. Cada local teve sua potência de sinal capturada cem vezes para cada mapeamento. Depois disso foram feitos os cálculos de posicionamento, baseados na potência de sinal capturada com o mapeamento. O mapeamento é o número de pontos por metro quadrado os quais foram utilizados como base para o cálculo.

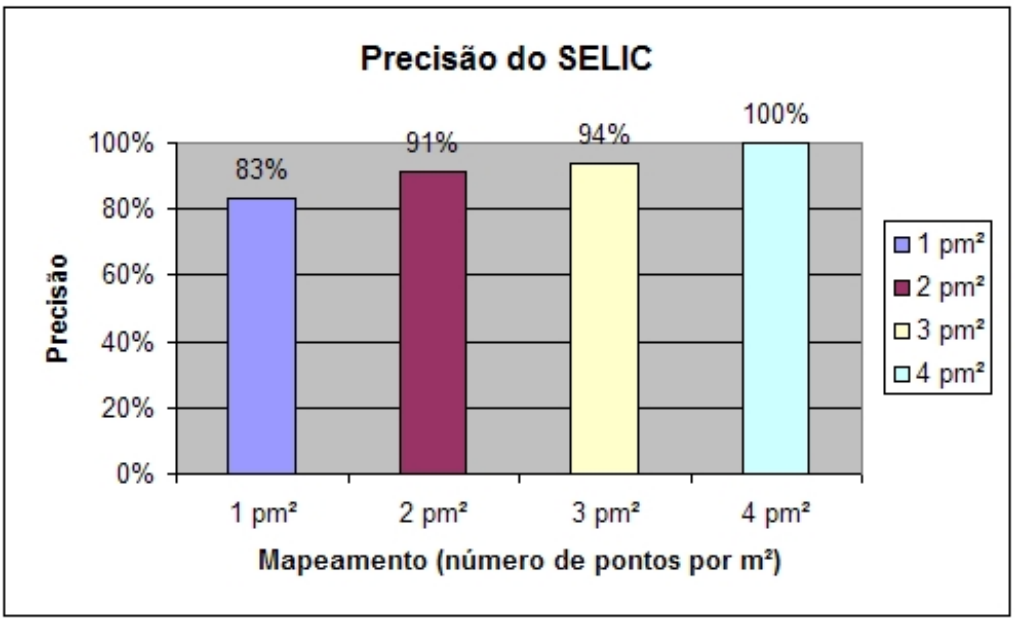

Figura 6.5 – Precisão do SELIC no segundo andar do prédio 6B da Unisinos

Na Figura 6.5 pode-se observar um aumento na precisão da localização conforme o número de pontos utilizados no mapeamento (eixo X). Isto deve-se ao fato de o cálculo conseguir encontrar uma maior quantidade de pontos para a potência de sinal da requisição, aprimorando a precisão do sistema.

6.1.3 Consumo de Energia

A utilidade de um dispositivo móvel está vinculada com a capacidade de sua bateria. Quanto maior a durabilidade da bateria, maior a probabilidade de utilização do dispositivo. Isso pode ser notado na evolução de dispositivos como celulares, *notebooks*, PDAs entre outros, que na última década receberam atenção especial para o projeto com processadores que possuem baixo consumo de energia. A durabilidade da bateria está vinculada também a utilização de serviços como Wi-Fi, *Bluetooth* ou programas como tocadores de música e vídeo.

O cliente do SELIC é um programa que procura utilizar uma quantidade

pequena de recursos do dispositivo móvel, procurando economizar ao máximo a bateria. Ambas as versões utilizam o sistema de localização IEEE 802.11 como padrão, sendo assim consultam quais as potências de pontos de acessos conhecidos e reportam ao *Web Service* do SELIC. Este processo é realizado com um intervalo de tempo pré-determinado, que pode ser ajustado pelo usuário.

Foi realizado um teste para constatar a durabilidade de baterias de dispositivos móveis utilizando o SELIC. Este teste tem o objetivo de verificar o consumo de bateria de um PocketPC quando está em execução o cliente do SELIC, validando se o sistema pode ser implantado na plataforma sem que tenha sobrecarga no consumo de bateria. Os equipamentos utilizados foram dez HP iPAQ linha hx4700 [1], disponibilizados pelo MobiLab, e que possuem as características descritas na Tabela 6.1. O teste foi realizado da seguinte forma para cada um dos equipamentos:

Descrição dos equipamentos	
Modelo:	HP iPAQ hx4700
Processador:	Intel PXA270 de 624MHz
Memória RAM:	64 MB
Bateria:	Íon de lítio de 1800 mAh
Tela:	4" VGA 64.000 cores
Rede:	IEEE 802.11b
Sistema Operacional:	Microsoft Windows Mobile 2003

Tabela 6.1 – Descrição dos equipamentos

- No primeiro momento as baterias foram carregadas até atingirem 100% de carga;
- O cliente do SELIC teve seu arquivo de configuração alterado para que a comunicação com o *Web Service* fosse de n segundos, sendo que n foi variado pelos valores: 15, 30, 45 e 60;
- O sistema foi inicializado registrando o horário da primeira requisição;

[1] Identificação dos dispositivos no sistema de patrimônio da Unisinos: 63901, 63902, 63893, 63888, 63897, 63877, 63905, 63878, 63907 e 63889.

- Quando o dispositivo teve sua bateria completamente descarregada, foi registrado o horário de desligamento;
- Foi calculada a média de durabilidade para a série de dispositivos usando o número de minutos entre o seu desligamento e o horário de inicialização da aplicação.

Com base na descrição do teste acima, foi realizada a média e o desvio padrão para cada valor de n, como podem ser observado na Tabela 6.2.

Tempo Coleta (seg)	**Média de duração bateria (h / min)**	**Desvio padrão (min)**
15	5,15 / 309	5,81
30	6,83 / 410	14,65
45	7,53 / 452	19,32
60	7,73 / 464	18,42

Tabela 6.2 – Consumo de bateria com o cliente do SELIC

Na Figura 6.6 é possível verificar a queda de durabilidade quando é utilizado um intervalo curto de tempo para a comunicação com o *Web Service* do SELIC. A comunicação com o *Web Service* utiliza XML e um *socket*. Cada vez que uma requisição é realizada ao *Web Service* há um consumo de processador, memória e rede. Quanto menor esse tempo, maior a quantidade de recursos necessários para o processamento. Outro fator de consumo excessivo de recursos é o "*parser*" de XML do *.NET Framework*, que utiliza bastante o processador e a memória.

Em ambientes onde o perfil de mobilidade dos usuários é contínuo, é aconselhável a utilização do sistema com tempo de aquisição de informações reduzido. Sendo assim, quanto menor o tempo de coleta de informações de antenas e comunicação com o *Web Service* do SELIC, maior é a precisão de localização. Em ambientes onde o usuário necessita de mais tempo de utilização do dispositivo e não possui mobilidade freqüente, por exemplo um professor em uma universidade, ou uma secretária num escritório, é indicado um tempo maior para coleta, evitando assim um consumo desnecessário de bateria.

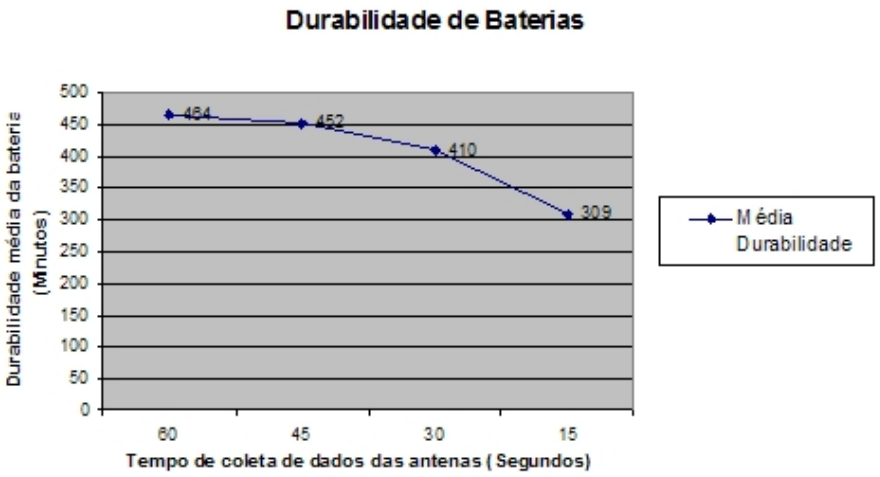

Figura 6.6 – Gráfico apresentando o consumo de bateria apresentado na Tabela 6.2

6.2 *Context Changer*

O primeiro passo para criar uma aplicação sensível à localização em Holo, é a criação da *HoloTree* baseada na árvore de localização dos ambientes, a qual encontra-se disponível no SELIC. Depois de criar o programa Holo com os entes representando os equipamentos e os locais, é necessário iniciá-lo no modo distribuído conectando-se ao HNS. Depois de iniciado o programa o HNS torna-se o responsável pelas modificações na *DHoloTree*, que serão realizadas usando as informações do SELIC.

O HNS é um sistema desenvolvido em C++. Ele mantém a estrutura da *DHoloTree* armazenada internamente, e realiza alterações conforme mensagens enviadas pelas HVMs. Toda vez que uma HVM processa uma instrução *move* ela envia uma mensagem para a porta UDP 9054 do HNS.

Para manter o sistema legado, o SELIC repassa as informações de localização dos equipamentos para o HNS no mesmo formato das HVMs. Foi utilizado o mesmo formato de mensagens para que não houvessem modificações no *parser* do HNS. Um exemplo da mensagem enviada pelo SELIC ao HNS é apresentado abaixo:

```
0008:barbosa:sala209
```

O demilitador para a mensagem é o caracter ":". Abaixo é apresentada uma descrição para cada item da mensagem:

1. O primeiro item identificado como "0008" é o código da requisição que trata da localização física no HNS;

2. O segundo item é o nome do equipamento, que pelo mapeamento identifica um ente no sistema;

3. O terceiro indica em qual local físico o equipamento se encontra.

Quando o HNS recebe uma requisição identificada como "0008", o *Context Changer* é encarregado de sincronizar na *DHoloTree* os entes que representem equipamentos. Esses entes são realocados para entes que representem locais físicos, com base nas informações do SELIC. Essa tarefa é realizada pelo *Context Changer*, módulo integrado ao *parser* da HNS. A Figura 6.7 apresenta trecho de código do módulo *Context Changer*.

```
01. switch (proto.getType()) {
02.   case ReportLocation:
03.      cout << "Informação de localização do SELIC..." << endl;
04.     cout << "Dispositivo " + proto.getBeing() + "..." << endl;
05.       tempdata = beingsList.searchData(proto.getBeing());
06.         if (tempdata != NULL) {
07.           cout << "Alterando o pai de" + proto.getBeing() << endl;
08.           tempdata->setBeingFather(proto.getBeingFather());
09.         } else {
10.           cout << "Ente não encontrado..." << endl;
11.         }
12.     break;
13. }
```

Figura 6.7 – Código para a mudança da *DHoloTree* pelo *Context Changer*

No SELIC existe um módulo chamado SELICHNS. Este módulo é o responsável pela comunicação com o HNS. Quando existem alterações dos equipamentos na árvore de localização, o SELICHNS informa ao HNS através de uma mensagem que o equipamento trocou de contexto.

Quando o HNS recebe uma mensagem do SELICHNS o código da Figura 6.7 é executado. Na linha 5 o *Context Changer* localiza o ente que recebeu na mensagem na estrutura *DHoloTree*. Se o *Context Changer* encontrar o ente (linha 6), é executado o procedimento para alteração do pai na linha 8. Caso o ente não seja encontrado, é então exibida uma mensagem para a saída padrão (linha 10).

6.3 *Holo Tree View*

A *Holo Tree View* (HTV) é uma ferramenta encarregada de apresentar o estado atual da localização dos equipamentos e/ou a *DHoloTree* do HNS. A prototipação foi realizada na linguagem de programação C# no ambiente de desenvolvimento *Microsoft Visual Studio 2005*, utilizando em conjunto, os aplicativos para criação de gráficos e árvores chamado *Graphviz* [55].

A HTV obtém informações de posicionamento dos equipamentos via *Web Services* do SELIC. A informação da *DHoloTree* é adquirida através de uma mensagem para o HNS, que retorna a informação formatada. Depois de receber as informações, elas são processadas para que a árvore receba o mesmo formato da taxonomia dos entes, proposta na seção 4.4. Após este processo, é utilizado o *Engine* de criação de árvores chamado "*dot*", que tem como resultado a imagem que é apresentada na interface do sistema.

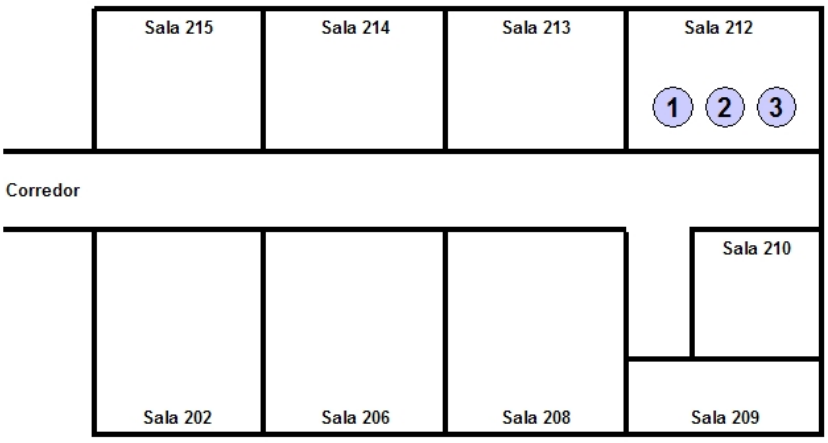

Figura 6.8 – Exemplo de posicionamento de dispositivos

A Figura 6.8 apresenta um possível cenário para a localização de três equipamentos ("*1*", "*2*" e "*3*") no contexto do MobiLab. Esses equipamentos estão sendo localizados pelo SELIC. Na Figura 6.9 é apresentado à árvore de contexto na *Holo Tree View* conforme o exemplo da Figura 6.8, depois da HTV conectar ao SELIC e receber as informações de posicionamento dos dispositivos. Conforme a proposta, os contextos são organizados de forma hierárquica.

A Figura 6.10 apresenta uma mudança de contexto para os equipamentos "*1*" e "*3*". A Figura 6.11 apresenta a situação da HTV após essas mudanças de contextos.

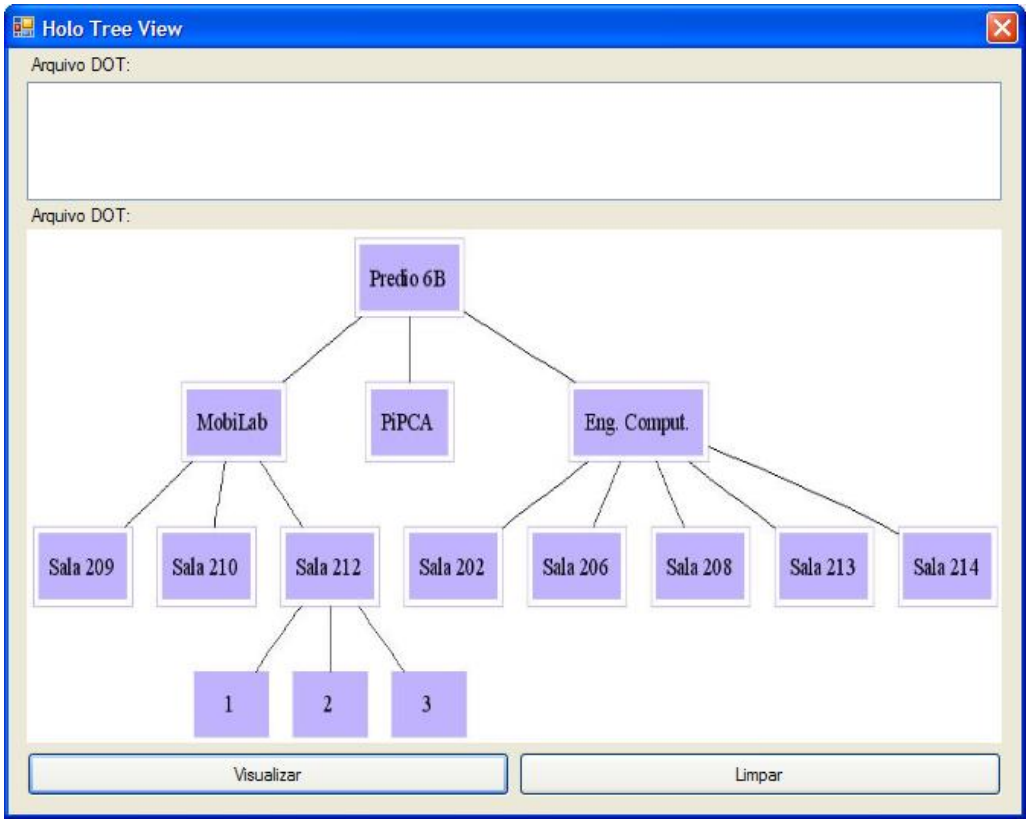

Figura 6.9 – Exemplo de saída da *Holo Tree View* para a situação da Figura 6.8

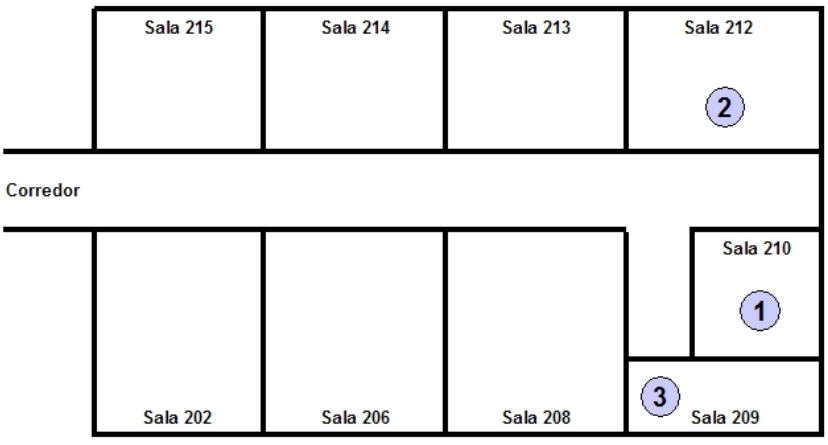

Figura 6.10 – Exemplo de mudança de contextos no mundo físico

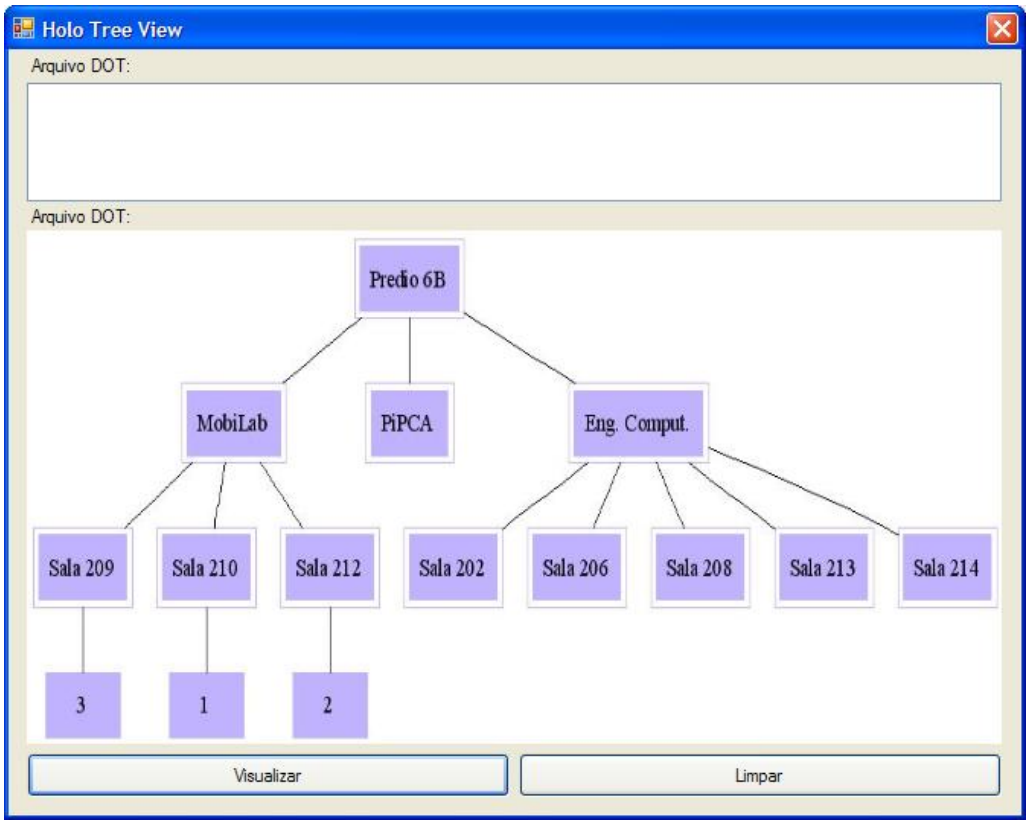

Figura 6.11 – Exemplo de saída da HTV para a Figura 6.10

6.4 Conclusões

Neste capítulo foram apresentados os aspectos de implementação do SELIC, *Context Changer* e *Holo Tree View*.

O SELIC foi desenvolvido usando o *Microsoft Visual Studio 2005*. Isso permitiu uma rápida prototipação. A portabilidade do SELIC também merece destaque. Atualmente o sistema é executado em ambientes *Windows* e *Windows Mobile*. Sua portabilidade para outros sistemas operacionais / arquiteturas depende também da portabilidade do *.NET Framework*. Foram realizados testes com o *Mono*, que é uma implementação do *.NET Framework* para o sistema operacional *Linux*, mas não foi possível executar o sistema.

O *Context Changer* foi integrado ao HNS sem a necessidade de adicionar novos tipos de mensagens no protocolo. A integração com o servidor de localização foi realizada através da inserção de um módulo no SELIC, denominado SELICHNS, que informa ao HNS quando ocorrem mudanças no posicionamento de dispositivos.

A *Holo Tree View* é a ferramenta para monitoração de ambientes baseada em

árvores de contexto. Atualmente encontra-se integrada ao SELIC e ao HNS, mas pode ser integrada com qualquer outro sistema.

No próximo capítulo serão apresentadas aplicações que utilizam o modelo HLS.

Capítulo 7

Aplicações

Neste capítulo serão abordadas aplicações que utilizam recursos do modelo HLS. Serão apresentadas duas aplicações que utilizam o SELIC como sistema de localização de dispositivos. Também é descrita uma aplicação criada para utilizar o ambiente de execução Holo para validação do modelo HLS.

7.1 LOCAL

A crescente procura de aplicações sensíveis à localização propiciou o surgimento de uma frente de pesquisa denominada aprendizagem ubíqua. O modelo LOCAL (*LOcation and Context Aware Learning*) [56, 57] usa informações de localização e de contexto para auxílio ao processo de ensino e de aprendizagem, e propõe o uso local (pequena escala) dos conceitos introduzidos por um modelo global de educação ubíqua (GlobalEdu) [58, 59, 60];

A arquitetura LOCAL é formada por seis componentes (Figura 7.1). O primeiro é um sistema de perfis de usuário, que armazena dados relevantes ao processo de ensino e de aprendizagem. O segundo é um sistema de localização (SELIC). O terceiro é um Assistente Pessoal (AP - Cliente SELIC) que acompanha o usuário, executando em seu dispositivo móvel. O quarto é um repositório de objetos de aprendizagem, que armazena e indexa o conteúdo relevante ao processo pedagógico. O quinto componente é um sistema de envio de mensagens contextuais e o sexto é um motor de análise (Tutor) que realiza inferências usando dados fornecidos pelos sistemas de perfis e de localização.

O Assistente Pessoal (AP) [58] é o módulo que acompanha o aprendiz

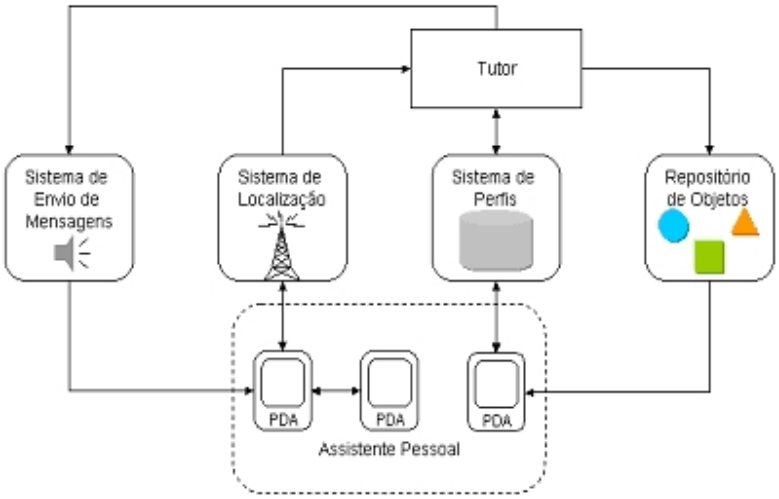

Figura 7.1 – Arquitetura do LOCAL

no seu dispositivo móvel. O AP possui as seguintes funcionalidades: (1) suporte a autenticação do aprendiz, ou seja, seu ingresso no sistema LOCAL; (2) armazenamento das informações persistentes do perfil; (3) suporte ao sistema de localização, permitindo o desligamento do mesmo se for de interesse do aprendiz; (4) suporte ao recebimento de avisos oriundos do sistema de mensagens. Atualmente o cliente do SELIC é integrado ao AP, ou seja, as funcionalidades do AP e do SELIC estão disponíveis no mesmo aplicativo.

O LOCAL utiliza a infra-estrutura de serviços do SELIC para enviar informações aos usuários. Esta informação é processada no Tutor, utilizando como base o *tracking* dos dispositivos no sistema e o perfil do usuário cadastrado no sistema. Outro recurso interessante é a utilização do sistema de envio de mensagens, onde um usuário autenticado no sistema pode solicitar que uma mensagem seja entregue em algum contexto, com data e horário definidos. Esse pode ser o exemplo de um professor, que informa ao sistema para que entregue o conteúdo da aula quando os alunos entrarem na sala de aula.

Uma sala do prédio do MobiLab (sala 206) foi utilizada como cenário, onde ocorreu uma aula simulada envolvendo dez aprendizes portando iPAQs 4700. A aula teve duração de duas horas e meia (entre 7:30 e 10:00). Os perfis dos alunos haviam sido previamente cadastrados no sistema. Os resultados do teste estão documentados na tabela 7.1. A tabela está organizada em cinco períodos considerados relevantes.

O primeiro período ocorre antes da aula. O professor agenda uma mensagem

com o tema de um debate que ocorrerá no encontro (no teste, "Paradigmas e Linguagens de Programação") e endereços na web contendo material relacionado. O registro durará por todo o período da aula e mesmo os alunos atrasados serão notificados. A mensagem está vinculada com o contexto (sala 206) onde ocorrerá o debate. O segundo período corresponde ao início da aula. Sete alunos entram na sala e são autenticados. Os alunos recebem a mensagem agendada. No terceiro período o professor solicita ao Tutor a criação de grupos para o debate. Os grupos são organizados (através de mensagens) de acordo com a similaridade nos interesses dos alunos por Linguagens de Programação. Neste caso, foram formados três grupos, um deles (grupo A) com quatro alunos interessados em Java. O outro (grupo B) com dois alunos com interesse em C# e o terceiro (grupo C) com apenas um aluno interessado em C++. No mesmo período, inicia o debate onde cada grupo apresenta as características das linguagens. Chegam dois alunos atrasados, que são autenticados pelo sistema. Ambos recebem a mensagem agendada. Além disso, são informados através de outra mensagem em quais grupos devem entrar. Um deles é encaminhado para o grupo A e o outro para o grupo C. No quarto período chega mais um aluno atrasado. Ele recebe a mensagem sobre o debate. No entanto, a atividade pedagógica já foi encerrada e o aluno não é encaminhado para nenhum grupo. O quinto período ocorre após o término da aula. O sistema registra as presenças dos alunos usando as informações de localização. Um aluno é considerado presente se esteve em aula pelo menos 60% do tempo. O aluno que chegou por último foi considerado ausente.

7.2 Agenda *Pervasiva*

A Agenda *Pervasiva*[1] não é uma aplicação para ser utilizada apenas em uma máquina ou em um dispositivo. Ela envolve um ambiente *pervasivo*, executando em conjunto com diversas máquinas, e possui suporte para atender inúmeros usuários utilizando diferentes tipos de recursos (por exemplo, localização e tocador de música).

[1] *Agenda Pervasiva* é tema de uma dissertação de mestrado do aluno Luis Henrique Ries, sob título "Uma Plataforma para Integrar Dispositivos Eletrônicos em Ambientes *Pervasivos*", no Programa de Pós-Graduação em Ciência da Computação, na PUCRS

Período	Horário	Personagem	Ações
1	07:00 – 07:30	Professor	Cadastra uma mensagem a ser enviada, no período entre 7:30 e 10:00, para todos os alunos que estiverem na sala 216.
2	07:30 – 08:00	Alunos	Chegam sete alunos (autenticação).
		Sistema	Sistema de Localização inicia o *tracking* dos sete alunos que chegaram. Sistema de Mensagens envia mensagens para os sete alunos.
3	08:00 – 09:00	Professor	Solicita ao Tutor a formação de grupos para o debate.
		Sistema	Envia mensagens para os alunos informando os grupos: Grupo A (quatro alunos, linguagem Java), Grupo B (dois alunos, linguagem C#), Grupo C (um aluno, linguagem C++).
		Alunos	Inicia o debate. Chegam dois usuários atrasados (autenticação).
		Sistema	Inicia o *tracking* dos alunos atrasados. Envia mensagem sobre o debate para alunos atrasados. Informa os atrasados sobre quais grupos devem participar: um deles entra no grupo A (Java) e outro no grupo C (C++).
4	09:00 – 10:00	Aluno	Termina o debate. Chega um aluno bastante atrasado (autenticação)
		Sistema	Inicia trace do aluno atrasado Envia mensagem sobre o debate O Tutor não envia mensagem de alocação de grupo, pois o debate acabou.
5	10:00	Sistema	Avalia o *tracking* dos alunos e registra as presenças.

Tabela 7.1 – Simulação de uma aula usando o LOCAL

A Agenda *Pervasiva* é uma aplicação distribuída que necessita de recursos para atender as suas funcionalidades e finalidades. Conforme foi mencionado, uma das finalidades dessa aplicação é atender inúmeros usuários permitindo o cadastro de seus compromissos e suas tarefas. A segunda finalidade é permitir alta interatividade com os usuários do sistema, disponibilizando diversos recursos para fazer o cadastro e a visualização de suas tarefas. Essa finalidade tem como proposta atender o requisito "onipresença". E por fim, o último objetivo é fornecer o rastreamento e a localização do usuário, para auxiliá-lo a relembrar de suas tarefas. A Figura 7.2 apresentada a arquitetura da aplicação.

A Agenda *Pervasiva* funciona da seguinte maneira:

1. Um usuário cadastrado no sistema insere compromissos especificando data, horário, descrição, mensagem e sala através de um terminal. A data e o horário representam a fatia de tempo que a tarefa deve ser realizada. A descrição representa o próprio compromisso. A mensagem representa uma mensagem de voz ou música que deve ser tocada no ambiente para lembrar o compromisso. A sala indica o local que o compromisso será realizado;

2. Um componente da aplicação chamado *Gerente* fica interagindo com os

terminais, efetuando o cadastro dos usuários e dos compromissos e atendendo todos pedidos dos usuários (por exemplo visualização de seus compromissos no ambiente). Esses cadastros e visualização são feitos através da interação do *Gerente* e do *Servidor de Perfil*, que contém os registro dos usuários e de seus compromissos;

3. Esse mesmo *Gerente* controla o tempo e todas as tarefas de todos usuários a fim de verificar quais os compromissos que estão sendo realizados e se estão sendo cumpridos pelos usuários;

4. O SELIC é reponsável por adquirir à localização de equipamentos, e contém informações relacionadas à localização de um determinado usuário (ou de seu dispositivo), quais os usuários que estão *online*, quais os usuários que estão em um contexto entre outros recursos;

5. Quando uma tarefa do usuário atinge o limite de tempo, o *Gerente* faz uma solicitação para o SELIC pedindo a localização do usuário;

6. O *Gerente* verifica no *Servidor de Perfil* se o usuário está localizado na sala em que o compromisso deve ser realizado. Se o usuário estiver nessa sala, o compromisso é cumprido e uma mensagem sonora (cadastrada na etapa 1) é tocada no ambiente.

7.3 Controle de segurança

O controle de segurança é uma aplicação escrita em Holo que tem a funcionalidade de ajudar vigilantes no controle de ambientes. O vigilante é uma pessoa que passa de sala em sala vasculhando o prédio para ver se existe alguém no recinto, verifica se o ar-condicionado está desligado e/ou as luzes apagadas. Após a verificação do estado do ar-condicionado e das luzes, o vigilante desliga ambos. A Figura 7.3 apresenta o código fonte da aplicação.

Esta aplicação tem o nome de *HoloGuard*. Para criá-la, primeiro foi necessário construir a *HoloTree* dos ambientes. Esses ambientes foram mapeados a partir da árvore de ambientes que está disponível no SELIC. Depois de projetar a árvore, foram adicionadas as seguintes ações para cada local do ambiente:

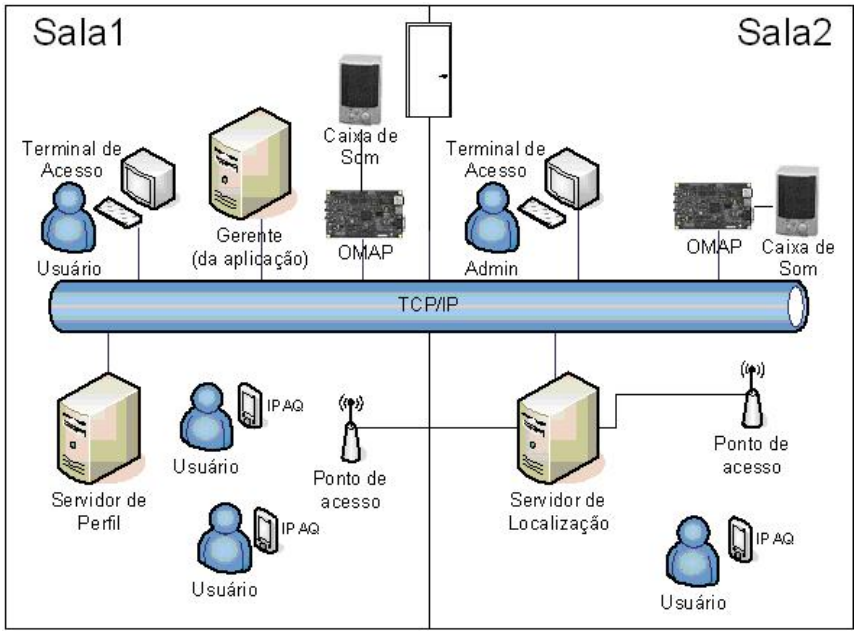

Figura 7.2 – Arquitetura da aplicação Agenda *Pervasiva*

1. *eac* (estado do ar-condicionado) - Informa qual a atual situação do ar-condicionado na sala, por exemplo: ligado ou desligado;

2. *el* (estado das luzes) - Retorna qual o atual situação das luzes no ambiente, por exemplo: ligado ou desligado;

3. *mac* (mudar ar-condicionado) - Altera o estado do ar-condicionado;

4. *ml* (mudar as luzes) - Altera o estado das luzes.

As linhas que começam com os caracteres // são comentários no código fonte. Um ambiente é definido da linha 40 até a linha 66. No ambiente são colocadas as ações que são relevantes ao contexto, e que serão disponíveis aos entes filhos. Essas ações estão definidas entre a linha 41 e a linha 66. Entre as linhas 67 até a 75 são realizadas colocadas informações para a história para definir o estado dos recursos.

A partir da linha 28 até a linha 39 é realizada a definição de um ente móvel. Ente móvel é o mapeamento de um dispositivo móvel do mundo real. A ação principal *movel* aguarda um *delay* na aplicação (linha 32) e pergunta quem é o ente pai do ente (linha 34). Na linha 36 é executada a ação *state*, que está presente no seu pai, na qual informa o atual estado dos recursos no ambiente.

A linha 3 é semelhante a função *void main()* na linguagem C, isto é, a primeira ação a ser executada em um programa Holo. No intervalo entre a linha 3 e a linha 25

```
01. holo(){
02.   holo(){
03.     clone(ambient, predio6b);
04.     clone(ambient, andar2o);
05.     move(andar2o, predio6b);
06.     clone(ambient, andar3o);
07.     move(andar3o, predio6b);
08.     clone(ambient, andar4o);
09.     move(andar4o, predio6b);
10.     clone(ambient, eng_comp);
11.     move(eng_comp, andar2o);
12.     clone(ambient, mobi_lab);
13.     move(mobi_lab, andar2o);
14.     clone(ambient, sala6b209);
15.     move(sala6b209, mobi_lab);
16.     clone(ambient, sala6b210);
17.     move(sala6b210, mobi_lab);
18.     clone(ambient, sala6b212);
19.     move(sala6b212, mobi_lab);
20.     clone(ambient, pipca);
21.     move(pipca, andar4o);
22.     clone(movel, fabiane);
23.     move(fabiane, sala6b212);
24.     clone(movel, cicero);
25.     move(cicero, sala6b209);
26.   }
27. }
28. movel(){
29.   movel(){
30.     while (){
31.       X := 10000;
32.       sleep(X); //Aguarda 10 segundos
33.       whoami(S); //Quem sou eu
34.       whereami(E); //Aonde eu estou
35.       writeln('Sou o ente ', S, ' estou no ambiente ', E);
36.       out(behavior)!state(); //Invocando a ação state do meu pai
37.     }
38.   }
39. }
40. ambient(){
41.   state(){
42.     eac(E);
43.     writeln('O ar-condicionado esta: ', E);
44.     el(L);
45.     writeln('As luzes estao: ', L);
46.     writeln('Desligando o ar-condicionado...');
47.     mac(desligado);
48.     writeln('Desligando as luzes...');
49.     ml(desligada);
50.   }
51.   eac(Estado){
52.     whoami(S);
53.     history?arcondicionado(S,#Estado);
54.   }
55.   el(Estado){
56.     whoami(S);
57.     history?luzes(S,#Estado);
58.   }
59.   mac(Para) { //altera ar-condicionado
60.     whoami(S);
61.     history!arcondicionado(S, Para);
62.   }
63.   ml(Para) { //altera luzes
64.     whoami(S);
65.     history!luzes(S, Para);
66.   }
67.   history{
68.     arcondicionado(sala6b209,ligado).
69.     arcondicionado(sala6b210,desligado).
70.     arcondicionado(sala6b212,ligado).
71.     luzes(andar2o,ligada).
72.     luzes(sala6b209,ligada).
73.     luzes(sala6b210,desligada).
74.     luzes(sala6b212,ligada).
75.   }
76. }
```

Figura 7.3 – Código fonte do *HoloGuard*

são realizadas as clonagens (semelhante a instanciação de um objeto no paradigma de orientação a objetos) dos entes e a montagem da *HoloTree*. A montagem a *HoloTree* é realizada pelo comando *move*. O comando *move* atribui um pai a um ente.

Depois de compilar o programa usando o compilador Holo, basta executar a aplicação na HVM utilizando a opção de execução distribuída, conectando ao servidor HNS, que receba mensagens do SELIC. A partir deste momento o estado da *HoloTree* será gerenciado com base nas informações de localização, ou seja, uma aplicação sensível a localização criada e executada no ambiente Holo.

7.4 Conclusões

Neste capítulo foram apresentadas aplicações que utilizam os recursos providos pelo modelo HLS.

Foram apresentadas três aplicações. O LOCAL e a Agenda *Pervasiva* são programas que utilizam os recursos do SELIC. A aplicação *HoloGuard* utiliza o SELIC e o ambiente de execução distribuída do Holoparadigma.

A criação desses programas provam a facilidade de integração do sistema. Todos os aplicativos foram desenvolvidos em plataformas diferentes. O LOCAL é desenvolvido em C# e PHP. A Agenda *Pervasiva* foi criada em C++. E o *HoloGuard* foi implementado no ambiente do Holoparadigma, utilizando o compilador holo, a HVM e o HNS. Essa heterogeneidade de ambientes de desenvolvimento provam a portabilidade e a facilidade de integração do modelo.

No próximo capítulo serão apresentadas as considerações finais deste trabalho.

Capítulo 8

Considerações Finais

8.1 Conclusões

Este trabalho apresentou uma proposta para suporte a aplicações sensíveis à localização no Holoparadigma. Esta solução engloba um servidor de localização (SELIC), um módulo para gerenciamento de entes denominado *Context Changer* e uma ferramenta para visualização de ambientes baseados em árvores de contexto.

No contexto do problema foram estudados quatro trabalhos relacionados, Aura, GAIA, one.world e PLACE de onde partiram as idéias para a formação de uma nova arquitetura, explorando os recursos e vantagens propostos pelo Holoparadigma. Com base nos trabalhos relacionados foram decididas as peculiaridades do modelo.

No SELIC destaca-se a interface de comunicação baseada em *Web Services*, o que permite integrar qualquer aplicação de forma simplificada. A modularização dos sistemas de localização no SELIC permite que sejam integrados novos sistemas sem a necessidade de alterar o protocolo com as aplicações (*Web Services*). Esta modularização também permite ao SELIC escolher o sistema de localização que tenha o maior grau de confiança na sua precisão, perante uma situação onde um equipamento é localizado por dois sistemas (por exemplo GPS para ambientes abertos e IEEE 802.11 para ambientes fechados).

Comparando o SELIC com os trabalhos relacionados, o mesmo tem a vantagem de sua interface ser baseada em *Web Services*, possibilitando a integração com qualquer aplicação, e não restringindo-o apenas ao ambiente do Holoparadigma. Esse recurso permite que a interface seja estendida pelas aplicações. Um exemplo de nova funcionalidade pode ser o controle de acesso por localização.

A adaptação automática da aplicação para programas em Holo sensíveis à localização é realizada pelo *Context Changer*. Esta é a principal diferença em relação aos projetos relacionados. O *Context Changer* recebe as informações do posicionamento dos dispositivos do SELIC, economizando a utilização de recursos de rede, processador e memória. Se cada dispositivo móvel que estivesse com a HVM em execução realizasse uma consulta para saber a sua localização física, a rede teria um *overhead* alto e o SELIC teria um problema de escalabilidade. Essa abordagem é possível porque atualmente o protótipo da HNS está sendo utilizado como um servidor central.

Em todos os trabalhos relacionados a manipulação da informação é realizada pela aplicação, que deve mudar o seu fluxo conforme o posicionamento do equipamento. Se existissem novos contextos no ambiente, será necessário recompilar toda a aplicação e adicionar novos *handlers* para cada novo local. Em Holo com a utilização da *HoloTree* é possível mapear todo o ambiente para uma árvore de contexto, tornando a aplicação altamente escalável. Quando novos ambientes são adicionados ao servidor de localização, basta adicionar os entes com suas ações na *HoloTree*, pois o tratamento da troca de contexto é realizado pelo ambiente distribuído (HNS).

A *Holo Tree View* é uma ferramenta que permite a um administrador o controle do ambiente onde tenha o SELIC e/ou o HNS. Com a HTV é possível identificar ambientes superlotados ou que estão congestionados no momento. A administração de contextos baseada em localização física só tem destaque especial no PLACE. O PLACE possui recursos de integrar o *tracking* de dispositivos baseados em informações de GPS, combinados com imagens de satélites. A *Holo Tree View* apresenta uma solução baseada em árvore de contextos, o que facilita a visualização da informação, e mantém a compatibilidade com a *Holo Tree* do Holoparadima.

Com a integração do modelo HLS ao Holoparadigma é possível afirmar que este ambiente de execução está mais preparado para o desenvolvimento de aplicações ubíquas.

8.2 Contribuições

Entre as principais contribuições deste trabalho destacam-se:

- Taxonomia para mobilidade de entes: Foi proposta uma taxonomia de entes, baseada nos seus comportamentos perante a mobilidade;

- Servidor de localização com precisão *Room Level*: Neste trabalho foi criado um servidor de localização que apresenta precisão a nível de sala, e que suporta múltiplos sistemas de localização. O acesso à API do servidor é realizada através de uma interface de *Web Services*;

- *Context Changer*: Integração da localização física dos dispositivos com o ambiente de execução do Holoparadigma;

- *Holo Tree View*: Uma ferramenta para administração de ambientes baseados em árvores de contexto;

- Teste de consumo de bateria: Neste trabalho foi apresentado um estudo sobre o consumo de bateria em dispositivos da arquitetura PocketPC, que utilizam uma aplicação cliente do SELIC. Nesse estudo foi possível observar o consumo de recursos dos dispositivos utilizando o SELIC, .NET *Framework* e *Web Services*;

- Teste de precisão do SELIC: Este teste comprovou a eficiência do SELIC na localização de dispositivos. O teste comprovou que o SELIC possui previsão *room-level*. Com esta precisão é possível criar programas sensíveis à localização para diversas áreas, como por exemplo hospitais, supermercados e escritórios;

- Aplicações: Foi desenvolvida a primeira aplicação escrita em Holo sensível à localização.

8.3 Trabalhos Futuros

Os seguintes trabalhos futuros poderão ser desenvolvidos no âmbito do HLS:

- Portar o cliente do SELIC para telefones celulares: A crescente utilização de celulares na última década trás um novo campo a ser explorado, onde o celular além de aparelho telefônico possui mais funcionalidades, por exemplo agenda, *player* de músicas e jogos. Estas funcionalidades podem ser melhor utilizadas quando integradas a um servidor de localização, por exemplo a busca

por usuários que estão jogando no ambiente, ou a utilização de uma lista de músicas (*playlist*) de outros dispositivos que estejam próximo;

- Integrar o suporte GPS no servidor de localização: Para aprimorar a precisão do SELIC em ambientes abertos é indicada a utilização de receptores GPS integrados ao servidor. A integração do GPS ao SELIC também trás a possibilidade de utilizar um sistema de coordenadas. Alguns exemplos da utilização de coordenadas são:
  - a aquisição da lista de dispositivos que estão a um raio x de uma coordenada;
  - a distância entre dispositivos;
- Integrar o SELIC na HoloCase: A HoloCase é uma ferramenta CASE para criação de programas Holo. Essa integração trará novos recursos e tornará mais simples o desenvolvimento de aplicativos sensíveis à localização;
- Ampliar os testes do servidor de localização: Testar a arquitetura do SELIC em outras plataformas, por exemplo, utilizando Linux e a plataforma Mono;
- Desenvolver novas aplicações: Criar novas aplicações que utilizem o SELIC ou o ambiente de desenvolvimento Holo. Novas aplicações trazem novos desafios e novas funcionalidades que podem ser integradas a interface do SELIC.

Bibliografia

[1] JUDD, G.; STEENKISTE, P. Providing contextual information to pervasive computing applications. In: *PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications*. Washington, DC, USA: IEEE Computer Society, 2003. p. 133. ISBN 0-7695-1893-1.

[2] RANGANATHAN, A. et al. Middlewhere: a middleware for location awareness in ubiquitous computing applications. In: *Middleware '04: Proceedings of the 5th ACM/IFIP/USENIX international conference on Middleware*. New York, NY, USA: Springer-Verlag New York, Inc., 2004. p. 397–416. ISBN 3-540-23428-4.

[3] BONATTO, D. T. *HNS: Uma Solução para Suporte à Execução Distribuída Considerando Aspectos da Pervasividade*. Tese (Dissertação de Mestrado) — Universidade do Vale do Rio dos Sinos, São Leopoldo, 2006.

[4] WEISER, M. The computer for the 21st century. *Scientific American*, Setembro 1991.

[5] FRANZ, D. *Exploração do Ambiente de Computação Móvel MHolo no Desenvolvimento de Aplicações*. Tese (Dissertação de Mestrado) — Universidade do Vale do Rio dos Sinos, São Leopoldo, 2006.

[6] BARBOSA, J. L. V. *Holoparadigma: Um Modelo Multiparadigma Orientado ao Desenvolvimento de Software Distribuído*. Tese (Doutorado em Ciência da Computação) — Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2002. 213p.

[7] BARBOSA, J. L. V. et al. Gholo: a multiparadigm model oriented to development of grid systems. *Future Gener. Comput. Syst.*, Elsevier Science

Publishers B. V., Amsterdam, The Netherlands, The Netherlands, v. 21, n. 1, p. 227–237, 2005. ISSN 0167-739X.

[8] BARBOSA, J. L. V. et al. Holoparadigm: a multiparadigm model oriented to development of distributed systems. In: *ICPADS*. [S.l.: s.n.], 2002. p. 165–170.

[9] BARBOSA, J. L. V. et al. Multiparadigm model oriented to development of grid systems. In: *International Conference on Computational Science*. [S.l.: s.n.], 2004. p. 2–9.

[10] MOBILAB. *Mobile Computing Lab*. Disponível em: http://www.inf.unisinos.br/m̃obilab/. Acesso em Junho de 2006.

[11] GARLAN, D. et al. Project aura: Toward distraction-free pervasive computing. In: . [S.l.: s.n.], 2002.

[12] SOUSA, J. P.; GARLAN, D. Aura: An architectural framework for user mobility in ubiquitous computing environments. In: *Proceedings of 3rd IEEE/IFIP Conference on Software Architecture*. [S.l.]: IEEE Computer Society, 2002.

[13] AURA, P. Acesso em março de 2006. Disponível em: http://www.cs.cmu.edu/ aura/.

[14] ROMAN, M.; CAMPBELL, R. H. Gaia: enabling active spaces. In: *EW 9: Proceedings of the 9th workshop on ACM SIGOPS European workshop*. New York, NY, USA: ACM Press, 2000. p. 229–234. ISBN 1-23456-789-0.

[15] ROMAN, M. et al. *A middleware infrastructure for active spaces*. Piscataway, NJ, USA: IEEE Educational Activities Department, 2002. 74–83 p.

[16] GAIA - Active Spaces for Ubiquitous Computing. Disponível em: http://gaia.cs.uiuc.edu/. Acesso em Junho de 2005. Disponível em: <http://gaia.cs.uiuc.edu/>.

[17] GRIMM, R. *System support for pervasive applications*. Tese (Tese de Doutorado) — University of Washington, 2002.

[18] MOKBEL, M. F. et al. Towards scalable location-aware services: requirements and research issues. In: *GIS '03: Proceedings of the 11th ACM international symposium on Advances in geographic information systems*. New York, NY, USA: ACM Press, 2003. p. 110–117. ISBN 1-58113-730-3.

[19] MOKBEL, M. F. et al. Place: A query processor for handling real-time spatio-temporal data streams. In: *VLDB: International Conference on Very Large Data Bases*. [S.l.: s.n.], 2004. p. 1377–1380.

[20] MOKBEL, M. F. et al. Continuous query processing of spatio-temporal data streams in place. In: *Workshop On Spatio-Temporal Database Management - STDBM*. [S.l.: s.n.], 2004. p. 57–64.

[21] MARKKULA, J. Dynamic geographic personal data - new opportunity and challenge introduced by the location-aware mobile networks. *Cluster Computing*, Kluwer Academic Publishers, Hingham, MA, USA, v. 4, n. 4, p. 369–377, 2001. ISSN 1386-7857.

[22] WIERENGA, J.; KOMISARCZUK, P. Simple: developing a lbs positioning solution. In: *MUM '05: Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*. New York, NY, USA: ACM Press, 2005. p. 48–55. ISBN 0-473-10658-2.

[23] TSAGKARIS, K.; DEMESTICHAS, P.; THEOLOGOU, M. Location- and service-aware downlink transmission power allocation in wcdma-based cellular networks. *Wirel. Pers. Commun.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 30, n. 2-4, p. 167–181, 2004. ISSN 0929-6212.

[24] MORAES, L. F. M. de; NUNES, B. A. A. Calibration-free wlan location system based on dynamic mapping of signal strength. In: *MobiWac '06: Proceedings of the international workshop on Mobility management and wireless access*. New York, NY, USA: ACM Press, 2006. p. 92–99. ISBN 1-59593-488-X.

[25] AL-QAIMARI, G.; FERNANDO, S. Location-aware applications: evaluating the ease of use and ease of learning. In: *IWCMC '06: Proceeding of the 2006*

international conference on Communications and mobile computing. New York, NY, USA: ACM Press, 2006. p. 1283–1288. ISBN 1-59593-306-9.

[26] GARZãO, A. S.; BARBOSA, J. L. V. Uma máquina virtual com suporte à concorrência, mobilidade e blackboards. In: UNIVERSIDAD MAYOR DE SAN ANDRéAS. *XXIX Conferência Latinoamericana de Informática (CLEI)*. La Paz, 2003. v. 24.

[27] BONATTO, D. et al. Estratégias para localização em um ambiente de computação móvel. In: *XXXII Seminário Integrado de Software e Hardware - SEMISH*. [S.l.: s.n.], 2005.

[28] W3C. *Web Services*. Disponível em: http://www.w3.org/2002/ws/. Acesso em Junho de 2006.

[29] NASCIMENTO, F. N. da C. *Um Serviço para Inferência de Localização de Dispositivos Móveis Baseado em Redes IEEE 802.11*. Tese (Dissertação de Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2005.

[30] HIGHTOWER, J.; BORRIELLO, G. Location systems for ubiquitous computing. *IEEE Computer*, v. 34, n. 8, p. 57–66, 2001.

[31] PADILHA, R. S. *DIGICLIP: Um Sistema de Localização e Rastreamento para o Ambiente Corporativo*. Tese (Dissertação de Mestrado) — Universidade Federal de Santa Catarina, Florianópolis, 2005.

[32] GARMIN. *Garmin: What's is GPS?* Disponível em: http://www.garmin.com/aboutGPS/. Acesso em Junho de 2006.

[33] GREUMANN, D. et al. Real-world implementarion of the location stack: The universal location framework. In: *WMCSA '03: Proceedings of the 5th IEEE Workshop on Mobile Computing System & Applications*. [S.l.]: IEEE Computer Society, 2003. ISBN 0-7695-1995-3.

[34] RASHID, O. et al. Extending cyberspace: location based games using cellular phones. *Comput. Entertain.*, ACM Press, New York, NY, USA, v. 4, n. 1, p. 4, 2006. ISSN 1544-3574.

[35] GRAJSKI, K. A.; KIRK, E. Towards a mobile multimedia age location-based services: A case study. *Wirel. Pers. Commun.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 26, n. 2-3, p. 105–116, 2003. ISSN 0929-6212.

[36] WANT, R. et al. The active badge location system. *ACM Trans. Inf. Syst.*, ACM Press, New York, NY, USA, v. 10, n. 1, p. 91–102, 1992. ISSN 1046-8188.

[37] SAVVIDES, A.; PARK, H.; SRIVASTAVA, M. B. The n-hop multilateration primitive for node localization problems. *Mob. Netw. Appl.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 8, n. 4, p. 443–451, 2003. ISSN 1383-469X.

[38] PRIYANTHA, N. B. et al. The cricket compass for context-aware mobile applications. In: *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 2001. p. 1–14. ISBN 1-58113-422-3.

[39] SMITH, A. et al. Tracking moving devices with the cricket location system. In: *MobiSys '04: Proceedings of the 2nd international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM Press, 2004. p. 190–202. ISBN 1-58113-793-1.

[40] NI, L. M. et al. Landmarc: indoor location sensing using active rfid. *Wirel. Netw.*, Kluwer Academic Publishers, Hingham, MA, USA, v. 10, n. 6, p. 701–710, 2004. ISSN 1022-0038.

[41] BAHL, P.; PADMANABHAN, V. N. RADAR: An in-building RF-based user location and tracking system. In: *INFOCOM (2)*. [S.l.: s.n.], 2000. p. 775–784.

[42] VARSHNEY, U. Location management for mobile commerce applications in wireless internet environment. *ACM Trans. Inter. Tech.*, ACM Press, New York, NY, USA, v. 3, n. 3, p. 236–255, 2003. ISSN 1533-5399.

[43] CHEN, Y.-C.; CHAN, Y.-J.; SHE, C.-W. Enabling location-based services in wireless lan hotspots. *Int. J. Netw. Manag.*, John Wiley & Sons, Inc., New York, NY, USA, v. 15, n. 3, p. 163–175, 2005. ISSN 1099-1190.

[44] BAHL, P. et al. Pawns: Satisfying the need for ubiquitous secure connectivity and location services. *IEEE Wireless Communications*, IEEE, 2002.

[45] PAHLAVAN, K.; XINRONG, L.; MAKELA, J. P. Indoor geolocation science and technology. *IEEE Communications Magazine*, IEEE, v. 40, n. 2, p. 112–118, 2002.

[46] ROOS, T. et al. A Probabilistic Approach to WLAN User Location Estimation. *International Journal of Wireless Information Networks*, Springer, v. 9, n. 3, p. 155–164, 2002.

[47] LADD, A. et al. Robotics-Based Location Sensing Using Wireless Ethernet. *Wireless Networks*, Springer, v. 11, n. 1, p. 189–204, 2005.

[48] YOUSSEF, M.; AGRAWALA, A.; SHANKAR, A. U. WLAN location determination via clustering and probability distributions. *Pervasive Computing and Communications, 2003.(PerCom 2003). Proceedings of the First IEEE International Conference on*, p. 143–150, 2003.

[49] HENGARTNER, U.; STEENKISTE, P. Implementing access control to people location information. In: *SACMAT'04: 9th ACM Symposium on Access Control Models and Technologies*. [S.l.: s.n.], 2004.

[50] HENGARTNER, U.; STEENKISTE, P. Protecting access to people location information. In: *SPC'03: Proc. of First International Conference on Security in Pervasive Computing*. [S.l.: s.n.], 2003.

[51] HAUCK, F. J. et al. A flexible and extensible object middleware: Corba and beyond. In: *SEM '05: Proceedings of the 5th international workshop on Software engineering and middleware*. New York, NY, USA: ACM Press, 2005. p. 69–75. ISBN 1-59593-204-4.

[52] BASTIDE, R. et al. Formal specification of corba services: experience and lessons learned. In: *OOPSLA '00: Proceedings of the 15th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*. New York, NY, USA: ACM Press, 2000. p. 105–117. ISBN 1-58113-200-X.

[53] ROCHA, R. C. A. da; ENDLER, M. Evolutionary and efficient context management in heterogeneous environments. In: *MPAC '05: Proceedings of the 3rd international workshop on Middleware for pervasive and ad-hoc computing*. New York, NY, USA: ACM Press, 2005. p. 1–7. ISBN 1-59593-268-2.

[54] BARBOSA, J. L. V. et al. Using mobility and blackboards to support a multiparadigm model oriented to distributed processing. In: UNB. *13th Symposium on Computer Architecture and High Performance Computing (SBAC-PAD 2001)*. Pirenópolis, GO Brasília, 2001. p. 187–194.

[55] GRAPHVIZ. *Graph Visualization Software*. Disponível em: http://www.graphviz.org/. Acesso em Junho de 2006.

[56] BARBOSA, J. et al. Mobile and ubiquitous computing in an innovative undergraduate course. In: *38th ACM Technical Symposium on Computer Science Education (SIGCSE)*. Covington, USA: ACM Press, 2007.

[57] BARBOSA, J. et al. Local: Um modelo para suporte à aprendizagem consciente de contexto. In: *XVII Simpósio Brasileiro de Informática na Educação (SBIE)*. Brasília, DF: SBC, 2006.

[58] BARBOSA, D.; GEYER, C.; BARBOSA, J. Globaledu: An architecture to support learning in a pervasive computing environment. In: *IFIP Workshop on Educational Technology (EDUTECH)*. Perth, Australia: [s.n.], 2005.

[59] BARBOSA, D. et al. Learning in a large-scale pervasive environment. In: *2nd IEEE International Workshop on Pervasive Learning (PerEL)*. Pisa, Italy: IEEE Press, 2006.

[60] BARBOSA, D.; GEYER, C.; BARBOSA, J. Uma proposta de agente pedagógico pessoal pervasivo - consciência do contexto e da mobilidade do

aprendiz. In: *XVI Simpósio Brasileiro de Informática na Educação (SBIE)*. Juiz de Fora, MG: SBC, 2005.