UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM COMPUTAÇÃO

FELIPE WEBER FEHLBERG

MultiS: um Servidor de Contexto voltado à Computação Pervasiva

Dissertação apresentada como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação

Prof. Dr. Cláudio Fernando Resin Geyer
Orientador

Porto Alegre, abril de 2007.

CIP – CATALOGAÇÃO NA PUBLICAÇÃO

Fehlberg, Felipe Weber

MultiS: um Servidor de Contexto voltado à Computação Pervasiva / Felipe Weber Fehlberg – Porto Alegre: Programa de Pós-Graduação em Computação, 2007.

91 f.:il.

Dissertação (mestrado) – Universidade Federal do Rio Grande do Sul. Programa de Pós-Graduação em Computação. Porto Alegre, BR – RS, 2007. Orientador: Cláudio Fernando Resin Geyer.

1.Computação Pervasiva. 2.Computação Consciente do Contexto 3.Linguagens de programação. I. Geyer, Cláudio Fernando Resin. II. Título.

AGRADECIMENTOS

Agradeço ao meu orientador, Cláudio Geyer por seu apoio e suporte desde que ingressei no mundo da pesquisa cientifica.

Agradeço a Luciano Cavalheiro, que tantas vezes me ouviu e me aconselhou em nossas reuniões com seu grande conhecimento sobre nossa área de pesquisa.

Agradeço especialmente à Iara Augustin, que me auxiliou em momentos mais decisivos do trabalho.

The reasonable man adapts himself to the world; the unreasonable one persists in trying to adapt the world to himself. Therefore, all progress depends on the unreasonable man.

George Bernard Shaw, 1903

SUMÁRIO

LISTA DE ABREVIATURAS E SIGLAS

API	Application Program Interface
Aura/CIS	Aura Contextual Information Service
CD-XML	Context Definition Extensible Markup Language
CIC	Composed-Info Cache
CSInt	Contextual Service Interface
DTD	Document Type Definition
EXEHDA	EXcution Environment for High Distributed Applications
IBM	International Business Machines
IP	Internet protocol
ISAM	Infra-estrutura de Suporte às Aplicações Móveis
J2EE	Java 2 Platform, Enterprise Edition
JCAF	Java Context-Aware Framework
MultiS	Multi-Sensor Context Server
PDA	Personal Digital Assistant
RMI	Remote Method Invocation
SGBD	Sistema de Gerenciamento de Banco de Dados
SQL	Structured Query Language
UFRGS	Universidade Federal do Rio Grande do Sul
UML	Unified Modeling Language
WiFi	Wireless Fidelity
WSAD	WebSphere Application Developer
XML	Extensible Markup Language

LISTA DE FIGURAS

LISTA DE TABELAS

RESUMO

A Computação *Pervasiva* tem sido tema de diversos trabalhos nos últimos anos. Essa emergente área de pesquisa propõe uma visão de futuro onde serviços computacionais são oferecidos para os usuários através de inúmeros dispositivos espalhados pelo ambiente. Os serviços são disponibilizados, tanto através da infra-estrutura existente dos computadores ligados fisicamente à rede quanto através de dispositivos móveis. Esse espalhamento da computação deve acontecer de maneira natural e imperceptível ao usuário. Dados pessoais, programas e arquivos de dados poderão ser acessados de qualquer lugar em qualquer momento. O poder de processamento será um recurso do ambiente, acessado quando necessário, da mesma forma que é hoje a eletricidade. O usuário não precisará ter ciência de qual máquina realiza o processamento necessário às suas aplicações, contanto que o resultado esperado seja obtido. Acredita-se que essa realidade será atingida através da aliança entre áreas de pesquisa como a Computação em Grade, Computação Móvel e a Computação Consciente do Contexto.

A Computação Consciente do Contexto busca enriquecer a comunicação entre os seres humanos e os dispositivos computacionais, tornando sua atuação mais eficaz. As aplicações conscientes do contexto conseguem perceber as modificações que ocorrem no ambiente e adaptar seu comportamento ao novo estado. Esse processo pode ser dividido em três etapas: monitoramento, reconhecimento de contexto e adaptação. Na etapa de monitoramento são coletadas, através de sensores, informações sobre o ambiente. Essas informações, entretanto, são geralmente, de baixo nível de abstração e, portanto, dificilmente usadas diretamente por aplicações. A etapa de reconhecimento de contexto relaciona os dados obtidos do ambiente e transforma-os para que possam ser úteis às aplicações no processo de escolha do comportamento mais adequado à cada circunstância, habilitando a etapa de adaptação a efetivar a transformação do comportamento da aplicação de acordo com a nova situação do ambiente.

Este trabalho propõe um servidor de contexto chamado MultiS que tem como objetivo a resolução dos problemas relativos à etapa de reconhecimento de contexto: a produção de dado de contexto baseado em informações de diversos sensores e a capacidade de reagir a modificações no ambiente. Também é proposta uma linguagem para composição de dados do contexto chamada CD-XML utilizada pelas aplicações para descrever ao servidor de contexto os dados aos quais elas são sensíveis.

Palavras-Chave: Computação Pervasiva, Computação Consciente do Contexto, adaptação, linguagens de programação.

MultiS: A Context Server for Pervasive Computer

ABSTRACT

The Pervasive Computing has been studied on several papers in the last years. This emergent research area presents a vision of future where computational services will be available through uncountable devices scattered across the environment. This service network will be exposed to the users by both traditional wired computers and mobile devices. This distribution of the computing is going to happen smoothly and transparently to the users. Personal data, computer programs, and data files will be available anywhere, anytime. The processing power will be an environment resource and will be accessed whenever needed, in the same way which is the electricity nowadays. The users will no longer need to worry about where their program is being executed, as long as he gets the needed result. The ISAM group believes that this new reality will be achieved through the alliance of research areas such as Grid Computing, Mobile Computing and Context-Aware Computing.

The Context-Aware Computing aims to enrich the communication between human being and computer devices. Context-aware applications are capable of recognize the changes on the environment and adapt its own behavior to the new context state. This process can be divided in tree steps: monitoring, context recognition and adaptation. On the monitoring layer, environment information is collected from sensors. Those sensors, however, usually return only low level information, which is hardly used by the applications on its original form. The context recognition layer processes the data acquired from the context and transforms into information aimed to be useful to the adaptation process. With that information the adaptation system can identify the correct behavior for the application on each different context situation.

This dissertation propose a context server named MultiS, which target is to solve the problems related to context recognition layer: the production of new context data based on the information of several sensors and the capability of react to changes on the environment. It also presents a new programming language for composition of contextual information, named CD-XML. This language is used by the context-aware applications to communicate to the context server describing which information the application is sensible to.

Keywords: Pervasive Computing, Context-Aware Computing, Adaptation, programming languages.

1 INTRODUÇÃO

A Computação *Pervasiva* tem sido tema de diversos trabalhos nos últimos anos. Essa emergente área de pesquisa propõe uma visão de futuro onde serviços serão oferecidos para os usuários através de inúmeros dispositivos espalhados pelo ambiente. Dados pessoais, programas e poder de processamento poderão ser acessados de qualquer lugar e a qualquer momento.

No cenário atual existe uma grande quantidade de aparelhos diferentes integrados através das redes de computadores. Houve uma popularização de dispositivos móveis como PDAs e celulares. O funcionamento atual desses dispositivos, no entanto, ainda não atingiu o estágio de amadurecimento que possibilite aproveitar os recursos existentes no ambiente. Parte importante dos esforços nessa direção reside na pesquisa da área da Computação Consciente do Contexto, que busca prover o ferramental necessário às aplicações, capacitando-as a adaptarem-se ao ambiente que as cerca.

O projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis Distribuídas) em desenvolvimento no instituto de informática da UFRGS tem por objetivo o desenvolvimento de uma infra-estrutura que ofereça as ferramentas necessárias à execução desse novo tipo de aplicação.

1.1 Motivação

No início de 2005, cientistas da The British Computer Society divulgaram uma lista do que acreditam ser os sete grandes desafios para a ciência da computação nos próximo vinte anos (BCS, 2005). Um desses desafios é a construção de sistemas ubíquos. Segundo esta lista, a Computação Consciente do Contexto, tema deste trabalho, tem destaque como uma das mais importantes propriedades da Computação *Pervasiva*.

Recentemente, muitos trabalhos científicos têm sido publicados com o intuito de promover a Computação *Pervasiva* (SATYANARAYANAN, 2001). Entretanto, os trabalhos já realizados focam, na maioria das vezes, aspectos específicos do contexto, como personalização ou independência de plataforma (KAPPEL, 2002). Outros projetos que buscam abordar contexto de maneira genérica, como o ContextToolkit (DEY, 2000-a), não abordam a questão da mobilidade (HOFER, 2003), de suma importância para a Computação *Pervasiva*.

A arquitetura de *software* ISAM (AUGUSTIN, 2001), diferentemente, tem a consciência do contexto como premissa de desenvolvimento não só das aplicações, mas também dos próprios serviços disponibilizados pelo *middleware* que gerencia as aplicações e o ambiente *pervasivo*. No ISAM, a consciência do contexto é responsabilidade do subsistema de Adaptação e Reconhecimento do Contexto (YAMIN, 2004). Porém, como concebido e implementado, esse subsistema apresenta importantes

limitações, destacando-se a questão de disponibilizar somente informações de contexto oriundas de um único sensor. Considerou-se que, na época, os *softwares* de aquisição e tratamento de informações de contexto enfatizavam o uso de uma única fonte de dados (mono-sensor). Segundo Laerhoven (LAERHOVEN, 2002), entretanto, o tratamento do contexto derivado de multi-sensores era um problema ainda a ser aprofundado.

O relacionamento entre dados de diferentes sensores acrescenta uma complexidade na interpretação do contexto pelas aplicações, já que existe uma variabilidade muito grande entre as possíveis características dos dados dos sensores. Dados estáticos ou dinâmicos, ou ainda, de diferentes domínios podem ser relacionados para a obtenção da informação de contexto desejada.

1.2 O Problema

Na Computação Consciente do Contexto, as aplicações adaptam seu comportamento de acordo com o ambiente e com modificações que nele ocorram. Para que essa adaptação seja possível, o sistema deve conseguir perceber essas transformações.

As informações sobre o ambiente são inseridas no sistema computacional através de sensores. Muitos deles, entretanto, fornecem informações de baixo nível de abstração que precisam ser processadas e transformadas em informação com significado agregado para, então, serem utilizadas pelas aplicações conscientes do contexto. Dados de diferentes sensores podem ser compostos para gerar uma única informação. Por exemplo, sensores que informam latitude, longitude e altitude podem, depois de processados, originar informações de alto nível sobre a localização de um determinado usuário (ie. se encontra-se em casa ou no trabalho).

Além disso, em um ambiente onde diversas aplicações estarão ativas ao mesmo tempo, como é a realidade prevista pela Computação *Pervasiva*, é provável que um mesmo dado de alto nível seja útil a mais de uma aplicação (CHEN, 2004-a). Atualmente, não existe nenhum sistema capaz de atender plenamente aos requisitos impostos por um ambiente com essas características.

O problema que se apresenta é, portanto, a criação de um sistema genérico de tratamento de informações de contexto capaz de compor novas informações a partir de inúmeros sensores. Dentro da perspectiva da Computação *Pervasiva*, diversas são as características que precisam ser atendidas pelo sistema de tratamento de informações de contexto: a solução deve oferecer facilidade de utilização, capacidade de reação às modificações no ambiente e, ainda, deve estar preparada para suportar inúmeras aplicações executando concorrentemente. Essas características são detalhadas no final do Capítulo 3.

1.3 Objetivos

O objetivo deste trabalho é apresentar uma solução para a aquisição e tratamento de informações de contexto, oriundas de multi-sensores. Como essa é apenas uma das etapas do processo de adaptação, a arquitetura deve ser capaz de ser integrada com outros *softwares* para seu completo funcionamento (outros subsistemas do ISAM oferecem alternativas nesse sentido). Diversos requisitos são necessários para resolver o problema supracitado.

O primeiro deles é a criação de um sistema de reconhecimento de contexto capaz de atender as necessidades das aplicações móveis, *pervasivas* e conscientes do contexto, conforme semântica estabelecida na arquitetura ISAM. O principal requisito desse sistema é a capacidade de gerar novos dados de contexto a partir de diferentes sensores e de disponibilizar essa informação às aplicações interessadas. É importante que o sistema esteja preparado para comunicar às aplicações as modificações que ocorrerem no ambiente, atendendo requisições do tipo inscrição/notificação. Além disso, a solução proposta deve ser facilmente extensível, possibilitando a inserção de novos mecanismos para criação e processamento de informações do contexto.

Outro requisito diz respeito à comunicação entre aplicações e o sistema de reconhecimento de contexto. Cada aplicação pode ter seus próprios requisitos sobre o contexto e pode compor informações de sensores de maneira própria. A descrição sobre como as informações dos sensores precisam ser compostas, para formar os dados de contexto de alto nível, precisa, portanto, ser repassada pela aplicação ao sistema de reconhecimento de contexto.

Esta comunicação precisa ser simples o suficiente para ser escrita por programadores de aplicação, que podem, muitas vezes, não ter conhecimento profundo da infra-estrutura de execução de aplicações. Para resolver esse problema, o segundo objetivo deste trabalho é o desenvolvimento de uma linguagem para transformação e composição de informações do contexto.

Resumindo, este trabalho está inserido dentro das atividades do projeto ISAM e visa aprimorar o Serviço de Reconhecimento de Contexto definido para o uso de multi-sensores, utilizando como suporte diversos trabalhos já realizados pelo grupo de pesquisa. Entre esses trabalhso encontram-se: a aquisição de informações dos sensores, que foi tema do trabalho PRIMOS (SILVA, 2003), a descoberta de recursos, que é tratada no trabalho PERDIZ (SCHAEFFER, 2005) e a disseminação das informações, que foi estudada (MORAES, 2005). Destaca-se o controle de acesso às informações produzidas, que está fora do escopo deste trabalho, informações sobre isso podem ser encontradas (HENGARTNER, 2003).

1.4 Estrutura do Texto

No Capítulo 2, discute-se a área de pesquisa onde o trabalho está inserido: Computação Ubíqua ou *Pervasiva* e Computação Consciente de Contexto. Além disso, analisam-se algumas diferentes definições do termo contexto. É apresentada também uma breve visão sobre o projeto ISAM, no qual este trabalho está inserido. No Capítulo 3 é revisado o estado-da-arte sobre Computação Consciente de Contexto. Diferentes publicações são analisadas sistematicamente. Posteriormente, são discutidas diversas características importantes dos sistemas de reconhecimento de contexto. Ao final do capítulo, analisam-se comparativamente os projetos estudados no que se refere às características previamente descritas. No Capítulo 4, é detalhado o MultiS, uma proposta de servidor de contexto capaz de combinar informações de diferentes sensores e disponibilizá-las a aplicações *pervasivas* conscientes do contexto. Como parte integrante da solução, apresenta-se também uma linguagem para composição de dados de contexto, chamada CD-XML. No Capítulo 5 são apresentados os detalhes de implementação do protótipo, assim como a integração desse protótipo à arquitetura ISAM/EXEHDA existente. Apresenta-se detalhadamente a utilização do MultiS em uma aplicação *pervasiva*, o PerMuseum. Ao final do capítulo são apresentados e

discutidos os resultados obtidos com testes de execução dessa aplicação. Finalmente, no Capítulo 6 são apresentadas as conclusões e são propostos alguns trabalhos futuros.

2 VISÃO GERAL DA COMPUTAÇÃO UBÍQUA

Neste capítulo serão apresentados os fundamentos da área de pesquisa na qual o trabalho está inserido. Inicialmente, se registra, brevemente, os fundamentos da Computação Ubíqua. Posteriormente, na seção 2.2, é discutida a Computação *Pervasiva*, considerada, pelo grupo, um estágio intermediário a ser atingido antes da Computação Ubíqua plena[1]. A Computação Consciente do Contexto provê ferramentas para a reatividade da Computação *Pervasiva*, e é apresentada na seção 2.3. Em seguida, estudam-se as principais definições do termo contexto. O capítulo encerra com uma apresentação resumida do projeto ISAM, escopo deste trabalho, enfatizando o subsistema de reconhecimento de contexto definido anteriormente.

2.1 Computação Ubíqua

A Computação Ubíqua é uma visão sobre o futuro da tecnologia e foi inicialmente proposta por Mark Weiser (WEISER, 1991). A concretização dessa visão, entretanto, ainda reside num futuro distante. No cenário proposto, computadores estarão totalmente integrados ao ambiente e à vida dos usuários, de forma que sua atuação não será percebida e sua utilização se tornará tão natural que será invisível aos usuários. Ao contrário do que acontece hoje, serão os computadores que deverão procurar a melhor forma de interagir com os seres humanos, adaptando seu comportamento de acordo com o ambiente no qual estão inseridos. Além disso, os inúmeros e diversificados dispositivos computacionais terão a possibilidade de interagir pró-ativamente entre si, formando uma única grande rede de disponibilização de serviços. A computação será um serviço como é hoje a energia elétrica. Vários projetos de pesquisa com esse tema estão em desenvolvimento, entre eles: Aura da Carnegie Mellon University (GARLAN, 2002-a), (GARLAN, 2002-b) e (AURA, 2005), Oxygen no MIT (OXYGEN, 2005), Endeavour em Berkeley (ENDEAVOUR, 2005) e Gaia na University Of Illinois (GAIA, 2005), (ROMÁN, 2002).

2.1.1 Invisibilidade (onipresença)

Em oposição à Realidade Virtual, onde as pessoas entram no mundo que existe apenas digitalmente, a proposta da Computação Ubíqua é que os computadores se integrem ao cotidiano dos seres humanos. Dispositivos com consumo reduzido de energia, ainda menores, com capacidade de comunicação poderão adaptar seu comportamento automaticamente de acordo com cada situação. Computadores com

[1] Muitos pesquisadores consideram a Computação Ubíqua e a Computação Pervasiva sinônimos.

comportamento autônomo não precisarão da atenção de seres humanos e passarão despercebidos. Segundo Weiser, no futuro, milhares de pequenos computadores poderão estar presentes dentro de uma única sala sem que se perceba sua presença.

No início do século XX, os motores eram visto com estranheza. Hoje, porém, não nos chamam mais a atenção. Não se pára para pensar que se está utilizando motores ao utilizar o vidro elétrico do carro ou ligar um aparelho de CD. Assim, também aconteceu com a energia elétrica que é praticamente onipresente em nossas vidas e com inúmeras outras tecnologias que foram assimiladas naturalmente. É provável que esse fenômeno se repita com a computação. Os computadores também devem ficar transparentes, e de certa forma, esse processo já se iniciou. Existem computadores, ainda que simples, em telefones celulares, caixas eletrônicos de bancos, carros, e em muitos outros lugares. Ainda assim, cada um desses dispositivos tem um comportamento isolado dos outros. É necessário que eles tenham a capacidade de se comunicar e perceber o meio para que possam oferecer a rede de serviços supracitada.

2.1.2 Situação Atual

A popularização dos dispositivos computacionais móveis é hoje uma realidade. Houve aumento no poder de processamento dos computadores, tempo de duração da bateria de PDAs e, mais recentemente, a integração de características computacionais em aparelhos celulares. Muitos deles são dotados de capacidade de armazenamento de dados e de poder de processamento. Alguns desses dispositivos possuem, inclusive, suporte à execução de código Java. As tecnologias de redes sem fio também tiveram uma evolução significativa nos últimos anos. Tecnologias como Bluetooth e implementações do padrão IEEE 802.11 são cada vez mais populares em dispositivos móveis.

Ainda assim, a Computação Ubíqua está longe de se tornar realidade. A computação ainda não é onipresente ao usuário final. Tecnologias importantes para sua concretização, como a computação de vestir (*wearable computing*), estão iniciando seu desenvolvimento. Problemas, como o consumo de energia, devem ser resolvidos. Além disso, os dispositivos ainda não oferecem os serviços de forma integrada (AUGUSTIN, 2004).

2.2 Computação *Pervasiva*

Uma visão da proposta de Weiser, mais próxima da realidade tecnológica, foi proposta pela IBM e recebeu o nome de Computação *Pervasiva* (*Pervasive Computing*). Segundo está proposta, o ambiente computacional estará espalhado entre os dispositivos, utilizando, tanto a infra-estrutura existente dos computadores ligados fisicamente à rede, quanto elementos móveis. Esse espalhamento da computação deve acontecer de maneira natural e invisível ao usuário (CHEN, 2004-a). Tanto dados pessoais dos usuários, programas e arquivos de dados, quanto o poder de processamento necessário às suas aplicações, poderão ser acessados de qualquer lugar e a qualquer tempo. Não será necessário que o usuário se preocupe com onde suas tarefas estarão sendo executadas, ou os dados foram armazenados, se no seu dispositivo móvel ou num grande servidor, contanto que ele tenha acesso ao serviço requisitado.

No cenário atual, tem-se uma grande quantidade de dispositivos diferentes comunicando-se através das redes de computadores. Dispositivos móveis como PDAs ou celulares são responsáveis por proporcionar ao usuário a mobilidade física, ou seja, a

capacidade de movimentar-se mantendo acesso aos recursos disponíveis. Além da mobilidade física dos dispositivos, os *softwares* deverão ser dotados de mobilidade lógica. A mobilidade lógica é a capacidade de componentes de *softwares* movimentarem-se entre dispositivos para prover um serviço de melhor qualidade ao usuário; isto é, um componente de *software* poderá migrar de um PDA para um computador com capacidade de processamento maior e, posteriormente, retornar o resultado da computação ao usuário, tudo isso de maneira transparente.

Finalmente, a consciência do contexto será responsável por detectar os recursos disponíveis no ambiente, possibilitando seu acesso pelos usuários e suas aplicações e adaptando as características das aplicações a essa disponibilidade. Segundo a visão do grupo ISAM, um ambiente de Computação *Pervasiva*, chamado Espaço *Pervasivo*, pode ser construído através da integração dos conceitos da Computação em Grade (*Grid Computing*), Computação Móvel e Computação Consciente do Contexto (YAMIN, 2004).

2.3 Computação Consciente do Contexto

A comunicação entre seres humanos é enriquecida por uma série de fatores implícitos, como expressão facial e entonação de voz. A comunicação que envolve computadores seja ela feita com outros computadores, seja com seres humanos, é, na maioria das vezes, explícita e carente de complementos que agreguem significado às informações. Essa comunicação poderia ser enriquecida com dados do ambiente em que os computadores estão inseridos, habilitando-os a ter uma atuação mais eficaz.

O contexto pode ser usado para interpretar e enriquecer a comunicação entre os seres humanos e os dispositivos computacionais. Em oposição ao método tradicional de entrada de dados, ele é utilizado implicitamente. Uma aplicação é consciente do contexto se é capaz de adaptar-se às condições do ambiente para melhor interagir com o usuário.

Um dos desafios da Computação Consciente do Contexto é interpretar a grande quantidade de informação disponível e conseguir determinar ações a partir da interpretação dessas informações. Pode-se, além disso, combinar informações provenientes de diferentes fontes (sensores) e fundi-las em uma nova informação com mais significado agregado (SCHMIDT, 1998).

Como os dispositivos móveis, utilizados no ambiente proposto pela da Computação *Pervasiva* têm, em geral, recursos limitados, é necessário otimizar seu uso através da utilização de recursos disponíveis no ambiente (fazendo uso de outros computadores com recursos de processamento disponíveis) (DEY, 2000).

Segundo (YAMIN, 2004) uma premissa da Computação *Pervasiva* é de que os usuários, ou o código da aplicação, ou ambos poderão se deslocar. A aplicação estará, portanto em diferentes contextos durante seu tempo de execução. Para que essas aplicações possam se adaptar às condições do ambiente em que se encontram é necessário que tenham acesso às informações sobre os recursos disponíveis. Esses dados devem estar estruturados com o objetivo de facilitar sua utilização pelas aplicações.

Atualmente o desenvolvimento de aplicações conscientes do contexto é uma tarefa complexa e trabalhosa. Essa situação pode ser remediada com desenvolvimento de uma infra-estrutura de suporte às tarefas comuns do desenvolvimento de aplicações

conscientes do contexto modelagem do contexto e tratamento das informações (HENRICKSEN, 2002).

2.4 Definições de Contexto

A definição de contexto tende a ser vaga, pois todas as coisas acontecem em determinado contexto (CHEN, 2000). Abaixo estão listadas algumas destas definições:

•O site Worldnet (WORLDNET, 2005) define contexto como o conjunto de fatos e circunstâncias que cercam um evento ou situação;

•Schimidt (SCHIMIDT, 1998) define contexto como o conhecimento relativo ao usuário e seu equipamento incluindo a situação que o cerca e sua localização;

•Dey (DEY, 2000) define contexto como o estado emocional do usuário, sua localização, inclusive no tempo, pessoas e objetos que o cercam;

•Zimmer (ZIMMER, 2004), define contexto como uma informação derivada de um sensor em conjunto com dados que são utilizados para interpretar o dado do sensor.

•Aurélio (AURELIO, 1999) apresenta contexto como: Encadeamento das idéias dum escrito; Aquilo que constitui o texto no seu todo; composição; Argumento, assunto.

Na visão do projeto ISAM, o contexto é definido como “toda informação relevante para a aplicação que pode ser obtida por ela” (AUGUSTIN, 2004). O contexto pode referir-se a informações ambientais (recursos físicos), funcionais (recursos lógicos) ou comportamentais (preferências e perfil do usuário).

2.5 Projeto ISAM

O projeto ISAM (Infra-estrutura de Suporte às Aplicações Móveis) tem como objetivo o desenvolvimento de uma infra-estrutura de suporte necessária para a implementação das aplicações móveis distribuídas com comportamento adaptativo em um ambiente da Computação *Pervasiva* (YAMIN, 2004).

A tese defendida é que a infra-estrutura de suporte à *pervasividade* em escala global pode ser construída através da integração de três áreas da computação: Computação Móvel, Computação em Grade e Computação Consciente do Contexto.

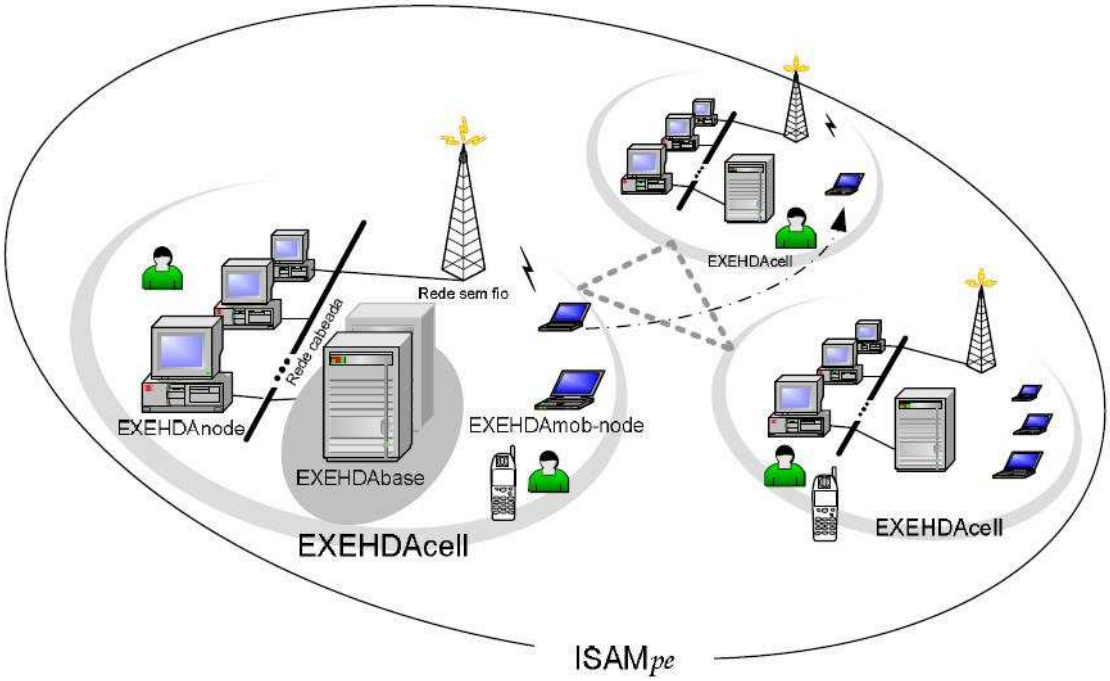

Figura 2.1: Organização celular da arquitetura ISAM (AUGUSTIN, 2004).

Os autores propõem uma infra-estrutura para execução de aplicações *pervasivas* onde o ambiente computacional é dividido em células, compostas por nodos móveis e por computadores pertencentes à rede cabeada tradicional. As informações relativas a cada contexto de execução estão disponíveis localmente em cada célula. Além disso, as células se comunicam através de um protocolo *peer-to-peer* (SCHAEFFER, 2004), possibilitando que informações sejam transmitidas entre células de execução. A Figura 2.1 mostra essa organização.

2.5.1 Arquitetura do Sistema

A arquitetura ISAM foi concebida para habilitar as aplicações a obter as informações do ambiente onde executam e adaptarem-se às alterações que ocorrem durante o curso da execução. Augustin (AUGUSTIN, 2004) propõe ainda uma linguagem de programação e um ambiente de desenvolvimento, especialmente desenvolvidos para atender às necessidades das aplicações da Computação *Pervasiva*.

Esta linguagem, chamada ISAMadapt, foi desenvolvida em conjunto com o ambiente de execução, chamado EXEHDA (*Environment Execution for High Distributed Applications*) (YAMIN, 2004) responsável por possibilitar a execução das abstrações do ISAMadapt. A linguagem de programação ISAMadapt teve sua origem no Holoparadigma (BARBOSA, 2002), acrescentando a esta proposta o suporte necessário à Computação *Pervasiva*. A arquitetura ISAM é composta por vários níveis, conforme apresentado na Figura 2.2. O trabalho desta dissertação se insere na parte relativa ao reconhecimento e tratamento dos dados do contexto, localizado na camada intermediária.

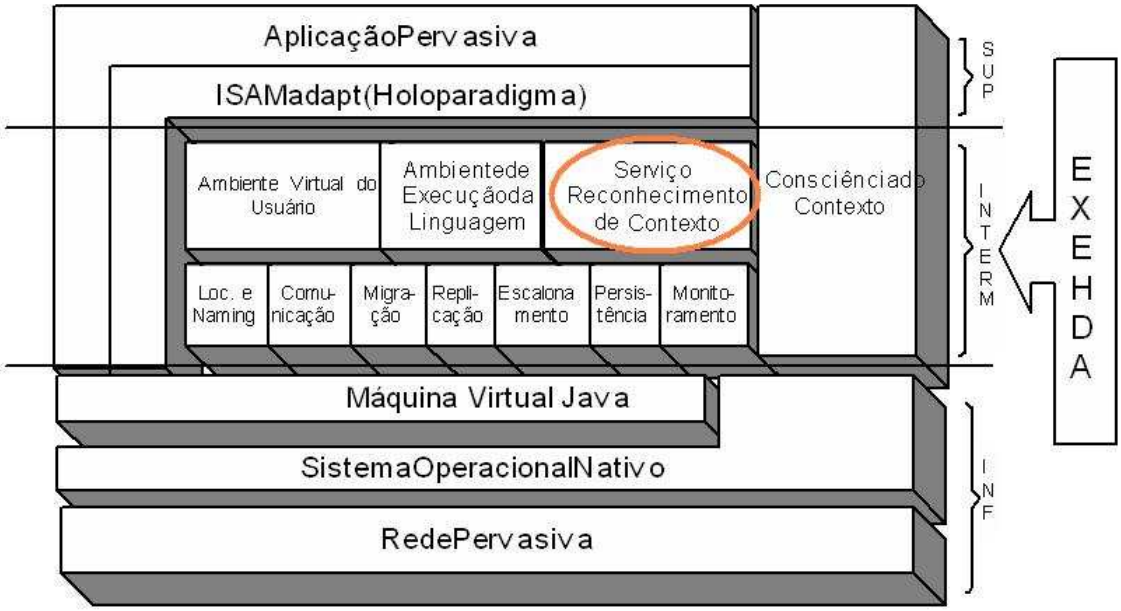

Figura 2.2: Arquitetura ISAM (YAMIN, 2004).

- A camada superior é a interface existente entre as aplicações e a infra-estrutura. Ela é composta da linguagem de programação ISAMadapt que faz uso de conceitos provenientes do Holoparadigma.
- A camada intermediária da arquitetura ISAM é composta pelo *middleware* EXEHDA. Esse módulo é responsável pela execução distribuída da aplicação e pelo gerenciamento dos recursos computacionais envolvidos. O *middleware* está organizado em diversos serviços, como o Serviço de Reconhecimento de Contexto, tema de trabalho, e o Serviço de Descoberta de Recursos.
- A camada inferior é composta pela linguagem de programação utilizada pelo *middleware* e pelo suporte de baixo nível como sistemas operacionais e infra-estrutura física. No protótipo atual do projeto ISAM foi escolhida, por razões de portabilidade, a linguagem Java.

Os recursos das aplicações são gerenciados pelo *middleware* EXEHDA. Para esse gerenciamento são utilizadas políticas de adaptação definidas pelo programador da aplicação durante o desenvolvimento. A responsabilidade sobre como executam as aplicações são, portanto, divididas entre o programador da aplicação e o *middleware* de execução. Para possibilitar esta divisão a linguagem de programação ISAMadapt e o *middleware* de execução EXEHDA foram desenvolvidos em conjunto, e estão fortemente acoplados.

2.5.2 Aplicações-Alvo

Na Computação *Pervasiva*, o usuário não enxerga os serviços divididos entre os dispositivos como na computação com perfil nômade de hoje. Ao contrário, o acesso às informações dos usuários e seu ambiente computacional, chamado Ambiente Virtual do Usuário, se dá através de qualquer recurso disponível, (*eg. desktops*, *notebooks*, PDAs, etc.) considerando-se as condições do contexto de execução. As aplicações-alvo são

distribuídas, tem mobilidade lógica e física e precisam ser adaptativas ao contexto em que estão inseridas.

O *middleware* oferece recursos para execução de aplicações *pervasivas*. Nesse sentido, a aplicação alvo do projeto tem características que possibilitam a mobilidade física, dos equipamentos e usuários. Além disso, as aplicações podem beneficiar-se da capacidade de mobilidade lógica de componentes de *software*. (*eg.* um componente é migrado para uma máquina com poder de processamento superior). Outra característica importante do sistema ISAM é a possibilidade das aplicações adaptarem-se às modificações do ambiente de acordo com características do meio ou recursos disponíveis. As aplicações-alvo são, portanto, adaptativas ao contexto.

Diferentemente do modelo nômade atual, onde as aplicações carregam dados de servidores e executam programas de sincronização. As aplicações no ISAM fazem uso de uma semântica siga-me. Os dispositivos móveis são a mais importante interface entre o usuário e o sistema. Eles não armazenam permanentemente nem código nem dados. Estas informações lhes são enviadas sob demanda.

2.5.3 Visão sobre a Definição de Contexto

O projeto ISAM tem uma definição prática do contexto. Segundo esta definição o contexto pode ser definido como "Toda informação relevante para a aplicação que pode ser obtida por ela" (AUGUSTIN, 2003).

No ambiente *pervasivo*, a mobilidade física introduz a possibilidade do movimento do usuário durante a execução da aplicação. Enquanto se movimenta, os recursos podem se alterar, não só em função da área de cobertura e heterogeneidade das redes, como em função da sua disponibilidade ser variável no tempo, devido à alta escalabilidade dos sistemas móveis distribuídos. Em conseqüência, a localização corrente do usuário determina o contexto de execução da aplicação. A Figura 2.3 esquematiza esta situação.

Figura 2.3: Contexto determinado pela mobilidade (AUGUSTIN, 2004).

A abstração de contexto permite focalizar em alguns aspectos que são relevantes em uma situação particular, enquanto permite ignorar outros. O programador explicitamente identifica as entidades e define seus atributos, os quais integram o contexto da aplicação (AUGUSTIN, 2003). Por exemplo, uma aplicação pode ter como elemento de contexto sensores que disponibilizam informações a respeito do poder computacional e da carga de processamento em dado momento. Alterações nos valores desses sensores disparam o processo de adaptação na aplicação. Assim, a definição de

contexto pode ser reescrita como “todo atributo de uma entidade cuja alteração em seu estado dispara um processo de adaptação na aplicação”.

Exemplos de informações que podem ser utilizadas pelas aplicações como dados de contexto são apresentados na Figura 2.4, que utiliza a metáfora de um guarda-chuva. Essas informações podem ser obtidas da parte fixa e móvel da rede, e de diversas fontes: sensores de hardware, sensores de *software*, informações sobre o perfil do usuário. Esses dados podem ser combinados, ou derivados de outras informações, e é a origem que dispara os processos de adaptação.

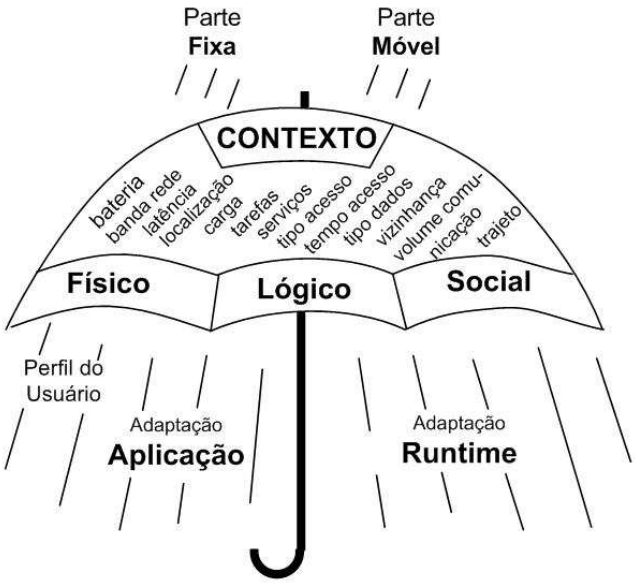

Figura 2.4: Exemplos de elementos de contexto ISAM (AUGUSTIN, 2004).

2.5.4 Planejando a Adaptação ao Contexto

O ISAM procura um conceito flexível de adaptação, o qual está relacionado com o contexto da aplicação. Assim, cada componente da aplicação pode determinar a que elemento(s) de contexto(s) é sensível, e quais de seus métodos têm um comportamento adaptativo.

A máquina de execução da aplicação (`AdaptEngine`) opera em conjunto com o serviço de reconhecimento de contexto numa relação baseada no modelo *publish-subscribe*. Quando é observado um dado de contexto correspondente a uma condição especificada por uma aplicação, um evento de adaptação é disparado.

A adaptação no ISAM ocorre em conjunto com a carga dinâmica de código de execução na aplicação. A aplicação, se adaptada à nova condição do contexto, passa a refletir o comportamento definido. Esta tarefa é realizada pela máquina de adaptação. Conforme a execução da aplicação progride, podem ocorrer novas adaptações.

2.5.5 Subsistema de Reconhecimento de Contexto e Adaptação

O processo de adaptação das aplicações ao ambiente está associado à execução do subsistema de reconhecimento de contexto, que inclui serviços que tratam do

monitoramento das informações brutas, passando pela identificação em alto nível dos elementos de contexto, até o disparo dos eventos de adaptação gerados em decorrência de uma alteração em um elemento de contexto. Esse subsistema, no EXEHDA, é composto pelos serviços `Collector`, `Deflector`, `ContextManager`, `AdaptEngine` e `Scheduler` detalhados em (YAMIN, 2004).

Augustin (AUGUSTIN 2003) apresenta, um esboço do sistema de reconhecimento do contexto. A estratégia inicialmente prevista para a produção de informações de contexto, consistia no emprego de cadeias de detecção de contexto, para cada contexto registrado, como ilustrado na Figura 2.5.

Estas cadeias seriam compostas por três elementos: *aggregator*, *translator*, *notifier*. O agregador (*aggregator*) seria responsável pela composição dos dados de um ou mais sensores, produzindo informação composta. O tradutor (*translator*) então transformaria o dado para um valor abstrato. Mudanças nesses valores abstratos seriam detectadas pelo notificador, gerando um evento de modificação de contexto.

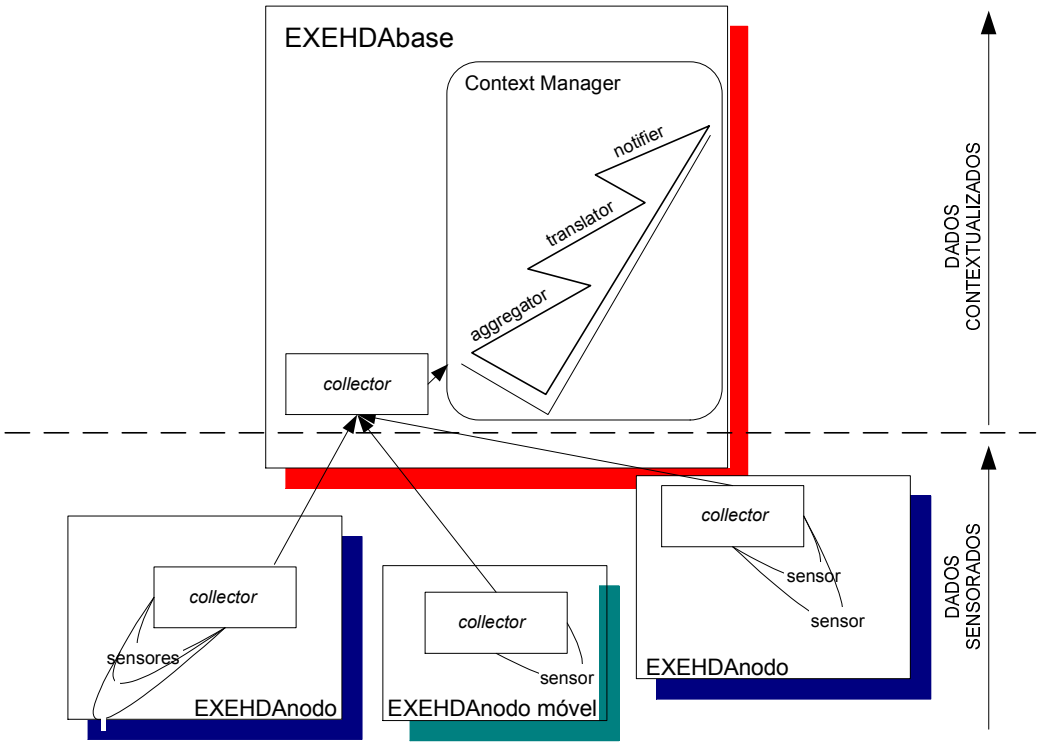

Figura 2.5: Tratamento de informações de contexto (FEHLBERG, 2005-b)

Esta dissertação estende e aprofunda esse modelo do servidor de contexto. As cadeias de detecção de contexto foram substituídas por árvores de inferência de contexto que permitem relacionar informações de diferentes sensores. Além disso, os três elementos responsáveis por tratar os dados de contexto foram substituídos pelo

conceito de filtro, mais genérico e flexível. Adicionalmente, define-se a linguagem CD-XML, utilizada pelas aplicações para especificar a formação dos dados de contexto.

3 INFRA-ESTRUTURAS PARA RECONHECIMENTO DO CONTEXTO

A realização de uma revisão no estado da arte dos *middlewares* para reconhecimento do contexto da Computação *Pervasiva* é o objetivo deste capítulo, no qual as principais publicações da área são analisadas de acordo com os mesmos critérios. Posteriormente discutem-se diversas características de grande importância para sistemas de reconhecimento de contexto. Ao final do capítulo apresenta-se uma comparação esquemática entre os projetos estudados no que se refere às características previamente citadas. Um estudo mais amplo sobre os trabalhos existentes na área pode ser encontrado em (FEHLBERG, 2005).

O primeiro projeto estudado é intitulado Solar, nele se propõe que o reconhecimento do contexto seja realizado por unidades de *software* chamadas operadores que podem ser compartilhados entre as aplicações e trabalham sobre fluxos de dados. Posteriormente é analisado o projeto Context Toolkit, trabalho reconhecidamente importante na área. Nele, assim como no Solar, as informações do contexto são tratadas de forma genérica. Aura/CIS é o nome do terceiro projeto apresentado neste capítulo. Sua proposta consiste em uma base de dados virtual que recebe consultas em uma linguagem similar à SQL e realiza, quando necessário, consultas a sensores ou outras bases de dados. Em seguida foi estudado o JCAF, proposta de um *framework* para reconhecimento do contexto. Finalmente o Egospaces, que propõem que o reconhecimento do contexto seja realizado por agentes móveis.

3.1 Solar

3.1.1 Histórico e Objetivos do Projeto

Solar é um projeto desenvolvido na universidade DartMouth no qual os autores propõem um *middleware* para tratar as informações do contexto de forma genérica e fornecê-las às aplicações. Segundo o modelo proposto, a infra-estrutura não é focada na obtenção da informação (*eg.* aquisição da informação junto aos sensores), sua ênfase é o tratamento e entrega de dados já processados para as aplicações.

O desenvolvimento do projeto é marcado pela implementação de dois protótipos. No primeiro deles, existia um nodo especial chamado *Star*, que centralizava as requisições das aplicações e instanciava novos nodos quando necessário. A comunicação então passava a ser feita normalmente entre os nodos sem a necessidade do nodo *Star*, ele apenas mantinha o registro dos nodos, aplicações do sistema.

A segunda versão do Solar evoluiu para uma arquitetura distribuída e auto-organizada (CHEN, 2004-b), sem a presença, portanto, do nodo centralizador. Na segunda versão do protótipo existem elementos que realizam o controle de acesso. O protótipo, nesta versão, conta com 13000 linhas de código Java.

3.1.2 Definição do Contexto e Aplicações-Alvo

O projeto objetiva ser um sistema genérico de suporte a tratamento e disseminação de informações de contexto para aplicações conscientes do contexto. Nele se introduz a proposta de uma grafo de operadores onde as informações do contexto são transformadas de acordo com as necessidades das aplicações.

O contexto é definido como a circunstância na qual a cada aplicação executa, e que pode ser caracterizada pelo ambiente físico, ambiente computacional ou pelo estado do usuário. Os dados de contexto são fornecidos por sensores que disponibilizam, através de fluxos dados, informações de baixo nível, que são posteriormente transformadas em informações de alto nível por operadores.

3.1.3 Arquitetura do Sistema

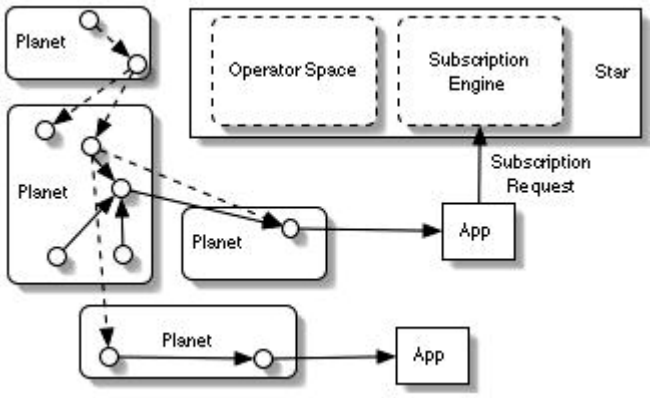

Figura 3.1: Arquitetura do projeto Solar (WHITE, 2002).

Na versão inicial, existiam dois elementos básicos de arquitetura, um nodo centralizado chamado *Star* e os demais nodos representando clientes, provedores e tratadores de informação com o nome de *Planets*, conforme apresenta a Figura 3.1. O nome do projeto deriva da analogia desta arquitetura centralizada com o nosso sistema planetário. O nodo *Star* é responsável pelo registro dos demais nodos e dos recursos do sistema. Como todos os outros nodos necessitam acessar o catálogo de informações freqüentemente, identificou-se um possível gargalo nesse elemento. Para resolver esse problema, o catálogo de informações foi distribuído entre os demais elementos, e é utilizado através de consultas do tipo *peer-to-peer*.

3.1.4 Serviços Fornecidos

3.1.4.1 Sensores e Coleta de Dados

Os sensores no sistema Solar registram-se em um nodo do sistema e publicam as informações através de um fluxo de dados. Na visão dos autores do projeto, os sensores

têm a visibilidade de apenas um subconjunto limitado de informações que descrevem o contexto de execução atual, sendo, portanto imprecisos. Uma informação de alto nível obtida com base nas informações de apenas um sensor pode, portanto obter interpretações incorretas. Utilizando-se diversos sensores para a geração destas informações de alto nível busca-se aumentar o grau de acerto das interpretações.

Os fluxos de dados processados pelos sensores geram eventos, que disparam o processo de interpretação das informações. Existem ainda alguns sensores que permitem consultas explicitas por parte das aplicações, neste caso, a requisição de informações de contexto também gera um evento no sistema.

3.1.4.2 Tratamento de Dados

O tratamento das informações geradas pelos sensores é realizado por elementos chamados operadores, que nada mais são do que *threads* localizadas em nodos do tipo *Planets* que filtram informações provenientes de um ou mais fluxos de dados.

Os operadores são unidades independentes de processamento de dados que se localizam nos nodos *planets*. Eles que utilizam pelo menos uma fonte de dados, seja um sensor, seja outro operador, executa um determinado processamento e fornece o dado processado a uma aplicação ou a outro operador. O processamento de uma informação de contexto pode passar por diversos operadores em diferentes *planets*. Existem quatro tipos básicos de operadores, entretanto outros podem ser acrescentados. Os quatro tipos básicos são: filtro, transformador, unificador e agregador.

3.1.4.3 Operadores

- Filtro: Esse operador, que é o mais simples existente, tem um fluxo de dados de entrada e um de saída. Sua funcionalidade é filtrar informações que não são interessantes à aplicação. Por exemplo, a aplicação pode estar interessada em ser notificada quando a temperatura de um processador atingir 90 graus. Mas pode não estar interessada em saber a temperatura caso esse valor não seja atingido. Neste caso, um operador filtro se encarrega de verificar o fluxo de informações do sensor e filtrar aqueles valores indesejados, informando a aplicação sempre que necessário.
- Transformador: Operador que tem um fluxo de entrada e um fluxo de saída. É utilizado para transformar um tipo de informação em outro. Geralmente de domínios diferentes. Ele pode ser utilizado, por exemplo, para transformar as coordenadas de latitude e longitude de um GPS para a localização física dentro de uma universidade (qual sala).
- Unificador: Pode ter diversos fluxos de entrada, mas tem apenas um fluxo de saída, em comparação com os dois primeiros tipos, é considerado um operador mais elaborado. Ele é responsável por combinar esses diversos sinais de entrada e gerar um resultado consistente no fluxo de saída. Esse operador pode ser especialmente útil quando usado para combinar fluxos de outros operadores. Pode-se ainda utilizar um operador unificador para a realização de funções lógicas (*and, or,* etc.). Esse tipo de operador poderia ser usado, por exemplo, par alertar a aplicação de um momento oportuno quando a cotação de uma determinada ação na bolsa está interessante e, além disso, o dólar está com um preço interessante (dois fluxos estão sendo utilizados como entrada: O preço da ação e a cotação do dólar).

- Agregador: Este é o operador mais elaborado existente no Solar. Ele tem diversos fluxos de entrada e leva em consideração um estado interno. Por exemplo, uma aplicação está interessada em saber quando a temperatura máxima do ano foi atingida em uma determinada região. Neste caso, a temperatura máxima já registrada até agora é o estado interno do operador.

3.1.4.4 Combinando Operadores

Os operadores podem ser conectados uns aos outros através da utilização do modelo *filter-and-pipe*, onde cada componente tem um conjunto de entradas e um conjunto de elementos de saída. Os elementos são associados conectando-se a saída de um elemento à entrada de outro.

Essa associação facilita a reutilização de componentes, pois cada operador realiza um tipo específico de processamento e pode ser instanciado para realizar esse processamento em outras situações com parâmetros de execução diferentes. Também é possível a reutilização de informações por outros elementos do sistema interessados na mesma informação (*eg.* outro operador ou uma aplicação). Além disso, esse modelo de comunicação facilita a distribuição dos elementos pela rede.

A disseminação das informações acontece através do grafo de operadores. Como já mencionado anteriormente, a associação entre os operadores baseia-se no modelo *filter and pipe*. Neste caso, o *filte*r são os operadores e o *pipe* são canais. Os canais são unidirecionais e possuem uma origem e um destino. Os sensores são fontes de dados (produtores) enquanto as aplicações são consumidores. Os operadores utilizam um ou mais sensores e entregam a informação processada agindo como uma nova fonte de dados.

Existem dois tipos de canais: *pull* e *push*. Nos canais *push* a informação é fornecida constantemente pelo sensor, enquanto nos canais tipo *pull* deve haver uma requisição explicita ao sensor para que a informação seja coletada.

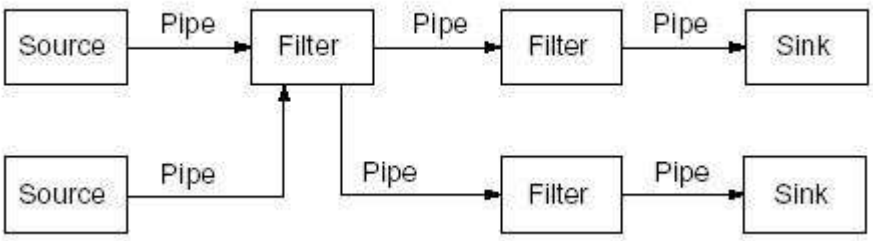

Figura 3.2: Composição de operadores (CHEN, 2004-a).

Os operadores podem ser combinados conectando-se a saída de um deles a entrada de outro, conforme mostra a Figura 3.2, desde que essa combinação respeite os tipos de canais, canais *push* só podem ser combinados com outros canais *push* e canais *pull* apenas com outros canais *pull*. Essas diversas combinações de sensores, operadores e aplicações, formam um grafo acíclico.

3.1.4.5 Disseminação dos Dados

Diversos canais podem estar conectados à saída de um operador. Uma solução simples para a disseminação destas informações seria enviar uma mensagem a cada um dos nodos. Esta solução, entretanto não é muito eficiente. Pode acontecer ainda que diversos operadores clientes estejam executando em um mesmo *planet*, que receberá diversas mensagens repetidas, uma para cada operador cliente da estrutura.

Para resolver esse problema, o Solar utiliza-se de um protocolo de comunicação *multicast* de nível de aplicação (*eg.* não *multicast* IP, pois esse pode ser filtrado). Esse protocolo implementa uma tabela de roteamento que diminui muito a proliferação de mensagens. A situação é ilustrada pela Figura 3.3.

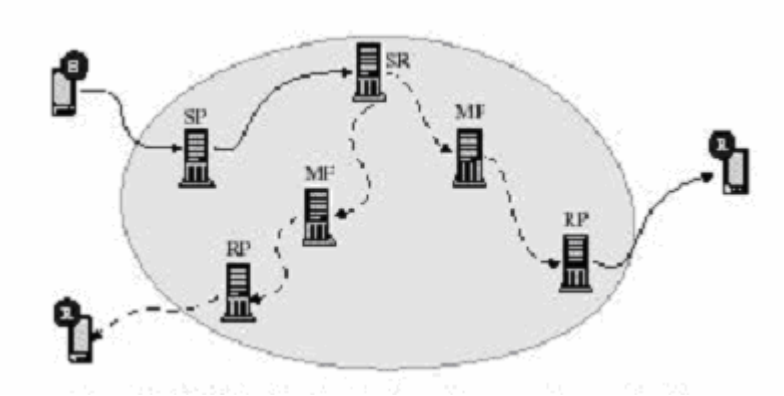

Figura 3.3: Difusão de informações no Solar (CHEN, 2004-a).

Na Figura 3.3, o nodo S, origem de um evento publica sua informação no *planet* ao qual está inscrito, SP. O *planet* atua como um *proxy*. A informação é difundida a partir de uma árvore *multicast* e as mensagens só são duplicadas quando necessário, de acordo com os caminhos da árvore (no exemplo da figura no nodo SR).

O grafo que especifica como as informações dos sensores devem ser tratadas para a obtenção desta informação de alto nível serão, portanto, iguais. O sistema pode então perceber que já existe um sub-grafo de uma outra aplicação que está realizando a tarefa que foi solicitada, suprimir sua instalação, e utilizar o sub-grafo já existente economizando recursos de infra-estrutura.

3.1.5 Acesso aos Dados pelas Aplicações

As aplicações clientes da infra-estrutura do Solar registram-se no sistema informando através de um arquivo XML, a especificação de como devem ser tratados os dados do contexto antes de serem entregues à aplicação. Essas informações são específicas de cada aplicação. Este arquivo XML é mapeado diretamente para um grafo de operadores da aplicação. Os operadores que ainda não se encontram em execução são instanciados para atender a esta nova requisição.

Aplicações distintas podem ter a mesma especificação para obter determinada informação, neste caso, os operadores do sub-grafo podem ser compartilhados conforme demonstrado na Figura 3.4. Essas semelhanças nos grafos, entretanto, são de difícil percepção pelo sistema. Com o objetivo de facilitar identificação de sub-grafos já existentes, os operadores que representam informações de alto nível, como a localização de determinado usuário, recebem nomes próprios. Quando acontece um registro de uma

nova aplicação, é realizada uma busca entre os nomes próprios de operadores já registrados no sistema buscando pontos comuns. Esta solução embora simplifique a tarefa da busca de operadores comuns, diminui a flexibilidade da solução, já que as aplicações precisam concordar com um nome único para o recurso a ser compartilhado.

A Figura 3.4 mostra como o compartilhamento de recursos pode ser importante em um sistema real. Os nodos cinza estão sendo compartilhados por pelo menos duas aplicações.

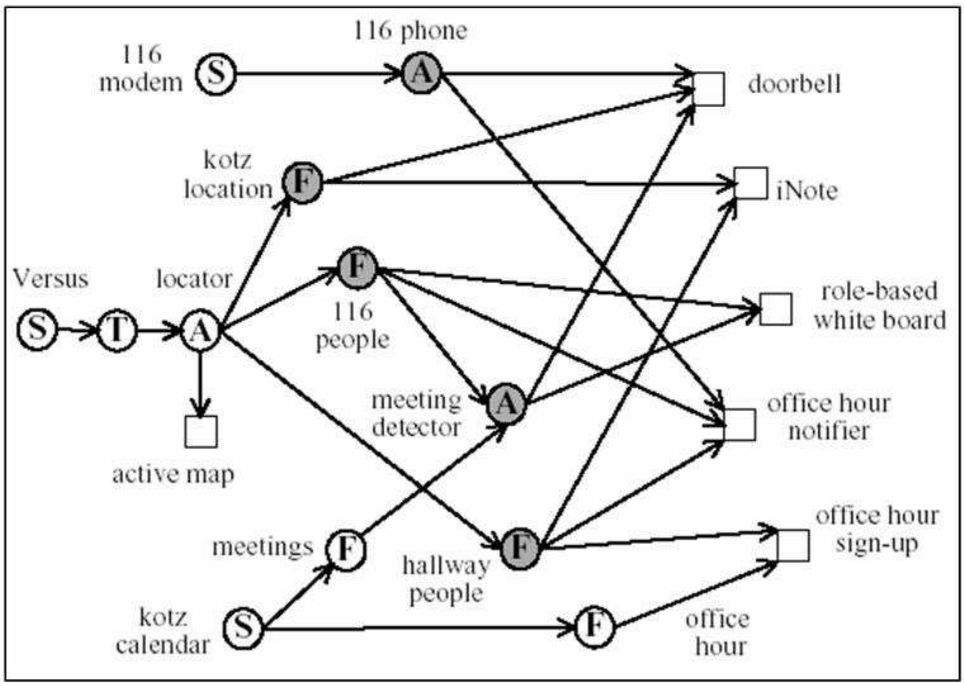

Figura 3.4: Compartilhamento de sub-grafos em uma situação de uso (CHEN, 2002)

3.2 Context Toolkit

3.2.1 Histórico e Objetivos do Projeto

Context Toolkit (DEY, 2000-b), desenvolvido na universidade de Berkeley, é um projeto onde se propõem um *framework* para implementação de aplicações conscientes do contexto, ele tem como principal objetivo facilitar o desenvolvimento desse tipo de aplicações através de mecanismos simples para aquisição, tratamento e entrega das informações do contexto às aplicações. Com a publicação de uma tese em 2000, o protótipo implementado conta com 29000 linhas de código Java. Além disso, muitos módulos foram desenvolvidos em outras linguagens.

O Context Toolkit é constituído de duas partes. Uma infra-estrutura de sistema, que suporta a execução de aplicações conscientes do contexto, e uma biblioteca de programação que auxilia o desenvolvimento desse tipo de aplicação. Outra contribuição do projeto é a proposta de uma metodologia de desenvolvimento de aplicações conscientes do contexto.

3.2.2 Definição do Contexto e Aplicações-Alvo

O contexto é tratado de forma genérica no Context Toolkit e é definido como qualquer informação que pode ser usada para caracterizar uma situação ou uma entidade. Uma entidade é uma pessoa, lugar ou objeto que é importante para a interação entre o sistema e o usuário, incluindo-se o próprio usuário.

Segundo essa visão, algumas informações de contexto são claramente mais importantes do que outras. O foco do Context Toolkit é responder às perguntas clássicas da Computação Consciente do Contexto: Quem? O que? Quando? Onde? Para tanto, informações como identidade do usuário e sua localização são preferidas. Afinal, dessas informações básicas, muitas outras podem ser derivadas Por exemplo, identificando-se o usuário é possível se descobrir seu telefone, suas preferências.

O projeto é direcionado às aplicações interativas que reagem à presença de usuários em um ambiente, não prevendo, portanto, as necessidades especiais das aplicações móveis. A especificação de como o contexto é utilizado é feita dentro do código da própria aplicação, mas contando com o auxílio do *framework* de desenvolvimento.

3.2.3 Arquitetura do Sistema

O Toolkit (DEY, 2000-a) descreve alguns componentes que formam a infra-estrutura de suporte às aplicações. Esses componentes são denominados `Context Widgets`, `Context Interpreters`, `Context Aggregators` e `Context-Aware Services`.

- **`Context Widgets`**: Representam os sensores do sistema e interagem com os dispositivos captando as informações de contexto e às disponibilizando ao *framework*, representam as entradas de dados no sistema.

- **`Context Interpreters`**: Realiza a interpretação do estado das informações de contexto, inferindo novos dados a partir dos captados dos sensores, gerando informações de mais alto nível de abstração.

- **`Context Aggregators`**: Unidades responsáveis por reunir diversas características de um mesmo elemento de contexto utilizando diversos `Widget` para isso.

- **`Context-Aware Services`**: São objetos que realizam ações que podem ser executadas pelas aplicações. Os `Context-Aware Services` são análogos aos `Context Widgets`, enquanto esses representam as entradas no sistema, os `Context-Aware Services` representam as saídas.

- **`Discoverer`**: Responsável por realizar a descoberta de recursos. Quando um `Widget, Aggregator ou Interpreter` é instanciado, um registro é feito no `Discoverer`. Este elemento é utilizado pelas aplicações para descoberta de recursos disponíveis no sistema, fornecendo os dados necessários à utilização desses recursos.

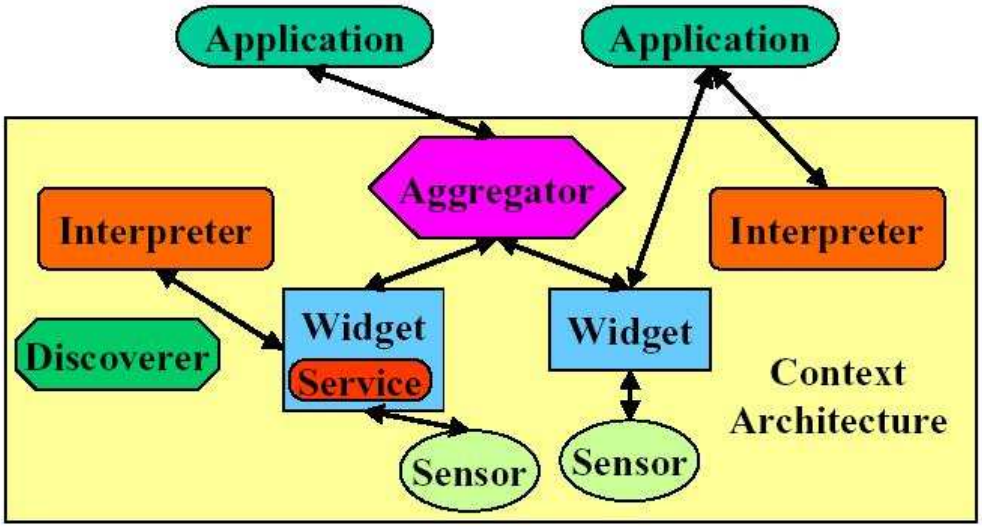

Figura 3.5: Arquitetura do projeto Context Toolkit (DEY, 2000).

A relação entre os elementos da arquitetura do projeto Context Toolkit é apresentada na Figura 3.5. Os componentes da arquitetura são disponibilizados através de um objeto Java chamado `BaseObject`. Esse objeto esconde das aplicações a lógica de distribuição e deve ser utilizado é para comunicação com outros elementos da infra-estrutura. No diagrama de classes da Figura 3.6, pode-se observar que esses outros elementos descendem de `BaseObject` . A comunicação entre os elementos é feita através de XML e HTTP.

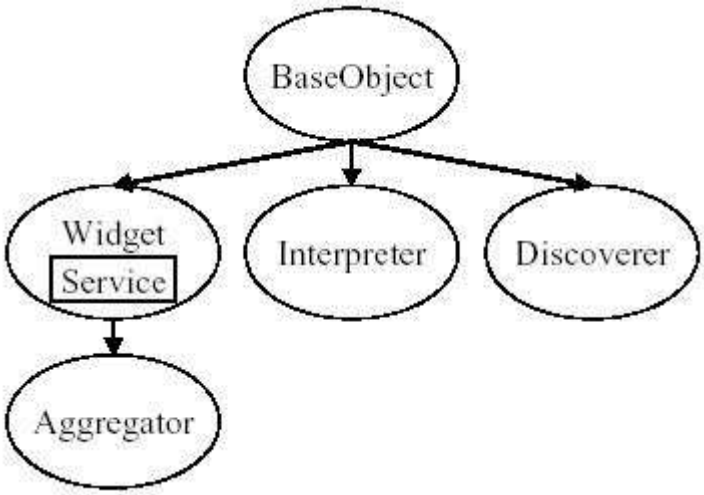

Figura 3.6: Hierarquia de classes de tratadores de dados no Context Toolkit (DEY, 2000).

3.2.4 Serviços Fornecidos

3.2.4.1 Sensores e Coleta de Dados

A tarefa de fornecer os dados do contexto às aplicações é feita por elementos chamados `Context Widgets` (o termo *widget* é tradicionalmente usado para

descrever componentes de toolkits de interface gráfica). Os `Context Widgets` são responsáveis por coletar as informações do ambiente, através dos sensores escondendo das aplicações a complexidade inerente a essa tarefa. Para implementar um novo `Widget`, o desenvolvedor deverá especificar os atributos de contexto que estarão disponíveis.

Esses componentes não pertencem a nenhuma aplicação, ao contrário, seu acesso pode ser compartilhado por diversas aplicações em execução. As informações são disponibilizadas com uma interface uniforme. Além disso, os `Context Widgets` são reutilizáveis.

3.2.4.2 Tratamento de Dados

Os `Context Widgets` fornecem informações brutas e de baixo nível de abstração. Duas estruturas são utilizadas para facilitar o uso de informações do contexto: `Context Interpreters` e `Context Aggregators`.

Os `Context Interpreters` são responsáveis por consultar os dados de `Widgets` e transformá-los, tornando-os úteis à aplicação. Por exemplo, um `Widget` fornece as informações sobre localização (latitude e longitude) de determinado usuário e um `Interpreter` realiza a conversão destas coordenadas em dados sobre a presença do usuário em determinado lugar.

Outro elemento utilizado para o tratamento do contexto é o `Context Aggregator`. Ele não modifica de fato nenhuma informação de contexto, sua função é reunir diversos tipos de informações de diferentes `Widgets` sobre um mesmo objeto. Os `Context Aggregators` são utilizados como mecanismo que facilita a implementação das aplicações, que podem acessar apenas um `Aggregator` abstraindo a consulta a diversos `Widgets`.

3.2.4.3 Disseminação dos Dados

O *framework* provê mecanismos que tornam o acesso aos dados distribuídos transparente na ausência de falhas. Os mecanismos distribuídos são acessados através de um objeto `BaseObject` que esconde da aplicação a lógica envolvida no processo.

3.2.4.4 Acesso aos Dados Pelas Aplicações

As aplicações podem se inscrever para serem notificadas em um evento de modificação de determinado contexto, ou realizar uma consulta explícita pelas informações do contexto diretamente. O acesso é feito através dos objetos disponibilizados na arquitetura.

3.3 Aura/CIS

3.3.1 Histórico e Objetivos do Projeto

Existe uma grande variedade de sensores responsáveis por alimentar os sistemas conscientes do contexto com informações relevantes. Esses sensores, entretanto, têm interface de consulta heterogênea. É necessária uma maneira de disponibilizar as informações de forma que seu acesso seja de fácil utilização pelas aplicações.

Uma alternativa é colocar as informações em uma base de dados, aproveitando a vantagem da padronização de acesso aos dados já existente hoje em aplicações que consultam bancos de dados e do isolamento da lógica e do peso de processamento dos dados nas aplicações.

Entretanto, bases de dados tradicionais teriam dificuldade em obter informações do contexto. Além disso, sensores muitas vezes têm meta-informações (como precisão e taxa de atualização) que não são diretamente suportadas pelas bases de dados.

O projeto (JUDD, 2003), desenvolvido na universidade Carnegie Mellon, propõe uma base de dados virtual que responda a uma linguagem de consulta similar a SQL, esta base virtual, porém realiza a consulta aos sensores sob demanda (*eg.* na hora da consulta). Pode haver *caching* de dados com o objetivo de aumentar a performance.

3.3.2 Definição do Contexto e Aplicações-Alvo

A estrutura projetada pode ser utilizada por uma variedade de tipos de informações do contexto. O projeto prevê inicialmente estruturas para representar três tipos de informações de contexto: pessoas, espaços físicos, dispositivos (*eg.* impressoras). Além disso, existe uma representação do contexto da rede (*eg.* taxa de utilização, histórico, etc.).

3.3.3 Arquitetura do Sistema

Na visão dos autores do projeto, a arquitetura tradicional de acesso a informações dos sensores é ineficiente, pois obriga cada aplicação a conhecer a lógica de acesso aos sensores. Além disso, a integração dessas informações é feita de maneira *ad-hoc* em cada aplicação. Essa situação é ilustrada na Figura 3.7. Os autores do projeto Aura/CIS propõem uma camada de *software* que abstrai a lógica de acesso e integração dessas informações. Segundo essa proposta, as aplicações consultam essa camada de *software*, chamada `Query Sinthesizer`. Para realizar essa consulta, é utilizada a linguagem CSInt (*Contextual Service Interface*), que tem forte influência da linguagem SQL.

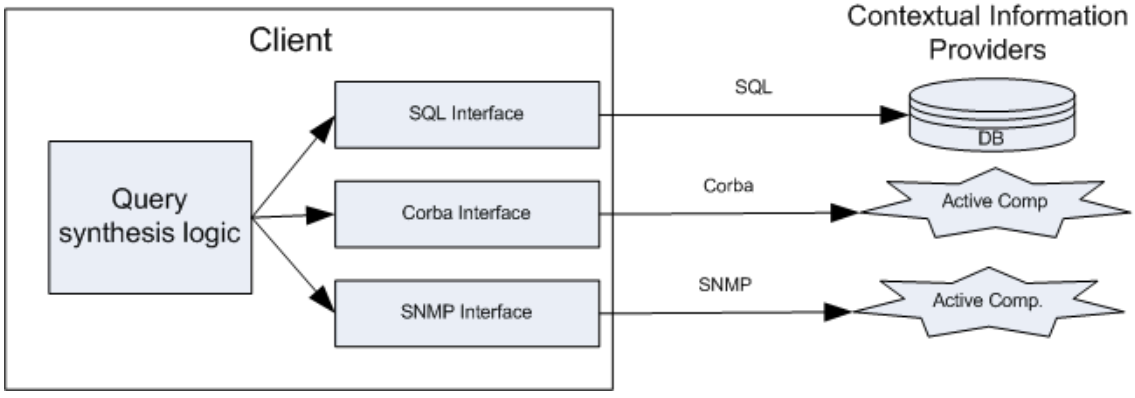

Figura 3.7: Arquitetura tradicional – complexidade de acesso a dados no cliente (JUDD, 2003).

O `Query Sinthesizer` decompõem a consulta da aplicação em diversas consultas de baixo nível que utilizarão também o CSInt. Estas consultas serão

repassadas a componentes que conseguem interpretar a requisição e respondê-la com dados dos sensores. Algumas informações de contexto são estáticas. Esse tipo de informação é armazenado em uma base de dados. A parcela da consulta relativa a itens que estão bases de dados pode ser repassada diretamente a essas bases, que já tem uma interface SQL. Outras informações são dinâmicas, e o custo de atualizar constantemente seu valor em uma base de dados pode ser muito alto. Portanto, são então utilizados componentes ativos que respondem diretamente às requisições (*ie.* a informação não é armazenada em uma base de dados). Destaca-se que a linguagem de consulta é a igual em ambos os casos, mantendo, portanto a consistência entre si.

Com essa estrutura as aplicações podem realizar consultas utilizando diversos dispositivos sem se preocupar com a forma de acesso dos dados ou com a decomposição das consultas ou sua recomposição.

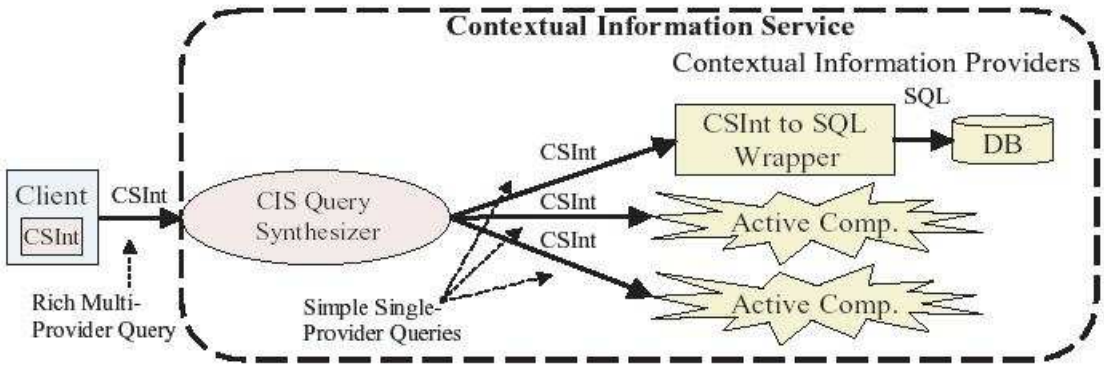

Figura 3.8: Arquitetura Aura/CIS – complexidade de acesso a dados no *middleware* (JUDD, 2003).

3.3.4 Serviços Fornecidos

3.3.4.1 Sensores e Coleta de Dados

No projeto Aura/CIS, os dados podem ser de dois tipos: estáticos ou dinâmicos. Dados estáticos podem ser armazenados em bases de dados tradicionais e sua consulta se torna bem simples, já que a linguagem CSInt preserva grande semelhança com SQL.

Dados dinâmicos são disponibilizados através de interfaces de consulta. Mesmo que a maneira tradicional de acesso a esse tipo de informação não seja uma linguagem similar a utilizada em bancos de dados. Esse direcionamento contribui para a homogeneidade do acesso às informações. Esses dados podem ser armazenados em *cache* para futuras consultas, desde que isso não deteriore a qualidade da informação.

3.3.4.2 Tratamento de Dados

Não existe uma estrutura específica de transformação dos dados como o `Aggregate` do Context Toolkit ou os operadores do Solar. Ao invés disto, esse sistema propõe que os dados resultantes dos provedores de informações de contexto (sensores) sejam tratados pelo `CIS Query Synthesizer`.

Não foram encontrados mecanismos para notificação de modificações do contexto. A falta desse mecanismo obriga a aplicação a realizar sucessivas consultas para perceber modificações no ambiente sempre que for necessário realizar uma validação desse tipo.

3.3.4.3 Disseminação dos Dados

A estrutura de acesso aos dados prevê apenas consultas explícitas aos dispositivos que fornecem informações de contexto. Os dispositivos são disparados assim que a consulta é realizada. Não existe na bibliografia analisada um suporte à disseminação de informações.

3.3.4.4 Acesso aos Dados Pelas Aplicações

As aplicações acessam os dados através de uma linguagem de consulta chamada CSInt, que é utilizada na aplicação é de alto nível e desconhece a forma de consulta dos dados. A é linguagem é similar a SQL com algumas adições para suportar consultas que contém os requisitos especiais das aplicações conscientes do contexto. Entre esses requisitos especiais estão: meta-atributos das informações (*eg.* precisão), incertezas, limites de tempo de resposta, etc.

```
Select APCoverage.room, APCoverage.apName
From Space, Device, DeviceLocation
      APCoverage, AccessPoint
Where Space.type = "Conference"
      and Space.ali within "ali://cmi/wean"
      and DeviceLocation.room = Space.name
      and DeviceLocation.id=Device.id
      and Device.type = "Projector"
      and APCoverage.room = Space.name
      and AccessPoint.apName = APCoverage.apName
      and AccessPoint.mbpsTotal < 1.0
Require
      mbpsTotal.sampleTime = start of meeting
      mbpstotal.sampleInterval = meeting length
TimeLimit none
```

Figura 3.9: Exemplo de consulta à CIS *Query Synthesizer* (JUDD, 2003).

Os dados são combinados de maneira genérica pelo `CIS Query Synthesizer`, eliminando a necessidade de implementação de código específico para combinar informações de mais de uma fonte de dados como acontecia no `Aggregator` do Context ToolKit.

3.4 JCAF

3.4.1 Histórico e Objetivos do Projeto

O JCAF (*Java Context Awareness Framework*) é um projeto que propõem uma infra-estrutura e uma biblioteca de programação voltada ao desenvolvimento de aplicações conscientes do contexto. Um destaque do JCAF é seu tratamento a

segurança, um ponto geralmente considerado importante, mas secundário em outros projetos de Computação Consciente do Contexto.

3.4.2 Definição do Contexto e Aplicações-Alvo

O contexto é tratado de forma genérica no projeto JCAF, e a definição de contexto é voltada às aplicações. Segundo (BARDAM, 2003) contexto são situações físicas e sociais nas quais os dispositivos computacionais estão envolvidos.

3.4.3 Arquitetura do Sistema

O ambiente de execução JCAF é baseado em diferentes serviços. Esses serviços comunicam-se através de *peer-to-peer*. Cada serviço, chamado `Context Service`, é responsável pela manipulação dar informações de contexto em um determinado ambiente (*eg.* um quarto, um escritório). A arquitetura do projeto pode ser vista na Figura 3.10.

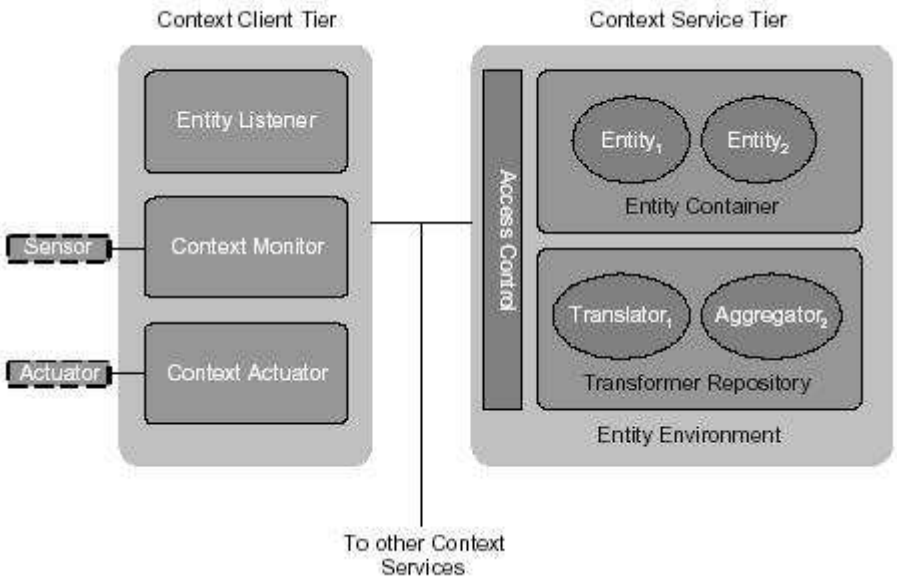

Figura 3.10: Arquitetura do projeto JCAF (BARDAM, 2003).

Um `Context Service` é um processo de longa vida similar a um `J2EE Application Service`. Esse processo abriga diversos objetos chamados Entity. Um `Entity` é um pequeno programa Java que é executado em um `Context Service` e responde a mudanças nos dados do contexto. O ciclo de vida do `Entity` é controlado pelo contexto ao qual ele foi adicionado. O `Entity` trabalha em conjunto com outros componentes do `Context Service` para cumprir suas tarefas. Para tanto, eles devem compartilhar recursos como conexões com bases de dados ou mesmo acesso a outros processos via RMI. O `Entity Enviroment`, análogo ao `Web Context` do J2EE, provê métodos para acesso a recursos compartilhados entre `Entitys`. Um tipo de recursos que pode ser compartilhado entre `Entitys` é o `Context Transformer.`

`Context Transformers` são programas Java específicos de determinada aplicação e são utilizados para transformar informações do contexto para esta aplicação.

Esses programas podem ser adicionados em um repositório de fontes chamado `Transformer Repository`. Esse repositório pode ser usado para carga dinâmica de dados. O componente `Access Control` autentica e autoriza requisições de aplicações clientes, realizando a tarefa de contribuir para a segurança dos dados do sistema.

Existem dois tipos de clientes desta arquitetura: `Context Monitor` (sensor) e `Context Atuator`. Clientes do primeiro tipo coletam informações do ambientes (contexto) cooperando com sensores, físicos ou não, e associa essas informações a um `Entity`. Um `Context Atuator` é responsável por interagir com o contexto, efetuando modificações em seu estado.

3.4.4 Serviços Fornecidos

3.4.4.1 Sensores e Coleta de Dados

Os `Context Monitor` são os elementos da infra-estrutura responsáveis por alimentar o sistema com informações de contexto. Essas informações são de baixo nível, sendo captadas diretamente de sensores, a lógica de acesso aos dados é oculta do resto do sistema. Os elementos não representam necessariamente elementos de hardware, informações lógicas também podem ser representadas por `Monitors`. As informações coletadas pelos `Context Monitors` são associadas a um `Entity` e desta forma disponibilizadas a toda infra-estrutura.

3.4.4.2 Tratamento de Dados

O tratamento dos dados do contexto se dá através de elementos chamados `Context Transformers`. Esses elementos registram-se para serem notificados quando ocorrer uma mudança em determinada propriedade de contexto, e uma vez que o evento aconteça, o elemento é ativado.

A especificação de um `Context Transformer` é feita através da implementação de uma interface Java. Dependendo dessa implementação, um `Transformer` pode atuar como um tradutor ou como um agregador de dados. No primeiro caso existe apenas uma fonte de dados, já no caso de uma agregador, diversas fontes de dados são consultadas para o processamento da informação. Em ambos os casos existe apenas um fluxo de dados retornado pelo `Transformer`.

Os `Context Transformers` ficam em um repositório de dados e sua carga é realizada dinamicamente. Esses elementos podem ser combinados em seqüência até que se crie a informação de contexto desejada.

Outro componente existente para o tratamento dos dados do contexto são os `Context Actuators`. Esses são elementos responsáveis por realizar a saída de dados afetando o contexto de execução. Da mesma forma que os `Context Transformers` eles são registrados e ativados apenas se determinada condição é atingida. Os `Context Actuators` podem ser úteis para manter a sincronia entre diversas informações de contexto (*ie.* a modificação de uma propriedade pode requerer a modificação em outra propriedade).

3.4.4.3 Disseminação dos Dados

O sistema foi projetado para execução em ambientes que não são amplamente distribuídos. Portanto a disseminação dos dados não foi um ponto tratado na bibliografia analisada.

3.4.4.4 Acesso aos Dados pelas Aplicações

As aplicações podem realizar consultas explícitas aos dados, chamadas assíncronas, ou podem se inscrever para serem notificadas sempre que um dado for modificado, nesse caso, são chamadas consultas síncronas. As chamadas assíncronas são implementadas através de subscrição em um `Entity Listener`, que detecta e sinaliza modificações em determinados `Entitys`. Esta monitoração pode ser feita em apenas um elemento de contexto ou em um conjunto de elementos do mesmo tipo.

A comunicação entre as aplicações e a infra-estrutura acontece através de RMI e utilizando os objetos da API de programação Java. Existe uma preocupação em desenvolver uma interface de utilização da infra-estrutura em *Web-Service* para que ela possa ser utilizada facilmente por outras linguagens de programação diferentes de Java.

3.5 EgoSpaces

3.5.1 Histórico e Objetivos do Projeto

O projeto EgoSpaces (JULIEN, 2002) consiste na proposta de agentes móveis de *software* que operam sobre componentes físicos de hardware visando adaptar seu comportamento de acordo com as mudanças no ambiente. Esses agentes interagem com o ambiente desempenhando tarefas previamente definidas quando determinada situação é alcançada, adaptando-se dessa forma a modificações nesse ambiente.

3.5.2 Definição do Contexto e Aplicações-Alvo

Os autores consideram que o contexto é sempre relativo ao observador, por isso, nesse projeto as informações são organizadas em visões. Cada agente define uma visão através da especificação de propriedades como sua localização etc. Uma visão é um subconjunto das informações acessíveis pelo agente, considerando apenas aquelas de seu interesse.

Dois usuários podem se referenciar a um objeto próximo, esse objeto pode ser diferente para cada um dos observadores, dependendo da localização do observador. A Figura 3.11 ilustra como dois usuários podem ter uma visão diferente sobre os dados do contexto dependendo de sua localização.

Um agente pode ter diversas visões e elas podem variar ao longo do tempo (com a movimentação do agente, por exemplo). O elemento de contexto que recebe uma atenção especial desse trabalho é a rede, suas topologia e elementos que podem ser encontrados na rede.

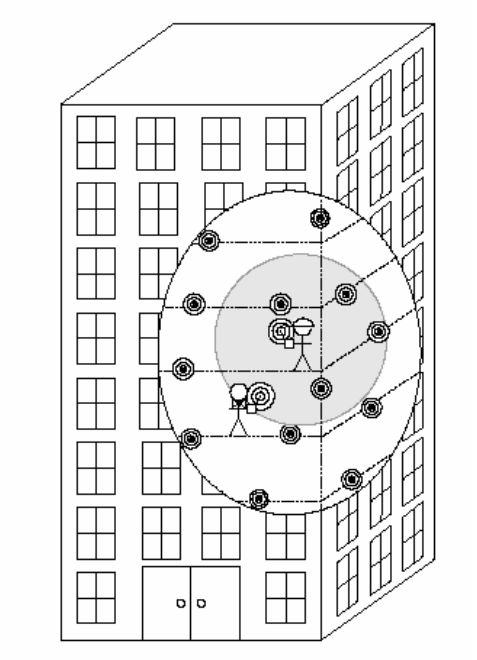

Figura 3.11: Contexto do usuário determinado por sua localização (JULIEN, 2002).

As informações de contexto são disponibilizadas no sistema através de um espaço de tuplas. Os autores do EgoSpace não parecem preocupar-se com a maneira como as informações são obtidas e inseridas no espaço de tuplas. Não foi encontrada na bibliografia pesquisada nenhuma referência a essa coleta. Esse espaço de tuplas é distribuído através dos agentes. Cada agente gerencia as informações de seu próprio espaço de tuplas.

3.5.3 Arquitetura do Sistema

A arquitetura do sistema não é o foco do EgoSpaces. Podemos entender como infra-estrutura os elementos propostos que são os agentes. O EgoSpaces se utiliza da infra-estrutura de outro projeto para comunicação entre os agentes, esse projeto é chamado LimeLite.

3.5.4 Serviços Fornecidos

3.5.4.1 Sensores e Coleta de Dados

O tratamento dos dados é feito através de agentes. Um agente no EgoSpaces é qualquer elemento de *software* com a capacidade de se movimentar através de *hosts* conectados. A especificação da visão é declarativa.

3.5.4.2 Tratamento de Dados

Programação reativa é utilizada para a ativação da adaptação dos agentes ao contexto. A ativação de um elemento acontece quando uma determinada situação é atingida nos dados representados no espaço de tuplas em uma determinada visão. Os elementos que são executados nesta situação são chamados `Behaviors`.

Uma tupla pode disparar apenas uma ação, sendo posteriormente consumida. Caso uma tupla seja responsável pelo disparo de mais de uma ação, apenas uma delas é executada. A escolha de qual ação será executada é não determinística.

3.5.4.3 Disseminação dos Dados

Uma das operações que podem ser realizadas quando acontece o disparo de uma ação é a migração de tuplas para outras visões. Através de um identificador único da tupla garante-se que apenas uma instância da tupla estará presente.

A operação de migração de tuplas é utilizada para a que os agentes tenham acesso transparente entre as tuplas. Como diversos agentes podem estar entrando e saindo da área de execução esta forma de comunicação é particularmente útil, já que o agente que tem interesse no dado pode solicitar a migração dele para sua área de atuação e operar sobre esse dado localmente.

Existe também a possibilidade de duplicação de tuplas. A operação na verdade cria uma cópia da tupla para que ela possa ser consultada localmente em casos de desconexão. A diferença entra a migração de tuplas e a duplicação é que na segunda, as duplicatas são apenas cópias e deu descarte não afeta a tupla original. O controle de duplicação infinita de tuplas deve ser feito por cada aplicação.

3.5.4.4 Acesso aos Dados Pelas Aplicações

As informações são disponibilizadas através de espaços de tupla. O *software* utilizado pelo EgoSpaces para a implementação do espaço de tuplas é o Linda. Os agentes podem abstrair a representação de dados em tuplas e utilizar outras estruturas de dados como árvores ou conjuntos. Existe uma camada de *software* chamada Venner que executa esse mapeamento.

3.6 Tópicos de Pesquisa e Tabela Comparativa

Durante o estudo dos *middlewares* existentes para reconhecimento de contexto na Computação *Pervasiva*, identificaram-se diversas características importantes que qualificam uma solução que se proponha a lidar com contexto genérico. Nenhum dos projetos estudados atende a todos os requisitos impostos pela Computação *Pervasiva*. Uma visão geral sobre a abrangência dos projetos é apresentada na Tabela 3.1 da seção 3.7. A seguir detalha-se cada uma das características relevantes para um *middleware* de reconhecimento de contexto. Posteriormente, discute-se a cobertura de cada uma das características pelos projetos estudados neste capítulo.

3.6.1 Compartilhamento dos Dados Monitorados

É contexto é percebido pela aplicação através de sensores em um ambiente. As aplicações podem ser sensíveis a diferentes variáveis (latência da rede, temperatura do ambiente, etc.). Em um ambiente real, portanto, devem existir diversos sensores, cada um deles especializado em obter um tipo de informação, para que o sistema obtenha informações e possa tomar ações em função das modificações no ambiente. Um *middleware* que suporte a execução de inúmeras aplicações deve conseguir fornecer os dados coletados para as aplicações interessadas sem que exista a necessidade de se coletar o dado novamente, evitando assim desperdício de recursos.

3.6.2 Compartilhamento dos Dados Tratados

Os sensores geralmente fornecem informações com pouco significado agregado. Ao relacionar dados de diferentes sensores ou transformar um dado segundo parâmetros descritos por uma aplicação, se obtém dados de mais alto nível. De sensores de temperatura, por exemplo, pode-se inferir se a temperatura de um ambiente encontra-se agradável para seres humanos. Muitas vezes, os tratamentos realizados sobre dados serão os mesmos para diversas aplicações, especialmente se elas estiverem no mesmo ambiente, pois elas tendem a utilizar os mesmos dados de alto nível. É interessante, portanto, que o sistema seja capaz não apenas de transformar os dados dos sensores, mas também fornecer esses dados a todas as aplicações que deles necessitam.

3.6.3 Independência entre Tratamento e Aplicação

Uma aplicação consciente do contexto é aquela que utiliza informações do ambiente buscando adaptar-se e reagir adequadamente às situações que se apresentam. Neste caso, existe claramente a opção de utilização ou não de um *middleware* de reconhecimento de contexto.

Se nenhum *middleware* estiver sendo utilizado, a tarefa de processar os dados do contexto, relacioná-los e modificá-los com o objetivo de tornar a informação mais útil será parte integrante da aplicação. Neste caso, cada aplicação precisará ter o conhecimento dos sensores existentes e de como a informação deve ser processada. Todas as aplicações que necessitarem de determinado tratamento precisarão ter, como parte integrante da própria aplicação, programas capazes de realizar o processamento necessário. O desenvolvimento de inúmeras versões de programas semelhantes aumenta o custo do desenvolvimento e as chances de introdução de erros nos programas desenvolvidos.

A utilização de um *middleware* para aquisição e tratamento de informações de contexto apresenta diversas vantagens sobre a primeira opção. É necessário desenvolver apenas uma vez cada componente, que pode ser então utilizado em diversas aplicações. O processo de teste e resolução de defeitos também é beneficiado, sempre que um defeito é corrigido, o é para todas as aplicações. Adicionalmente, os programadores de aplicação podem focar seus esforços na resolução de problemas específicos da lógica de negócios própria da aplicação, abstraindo funções secundárias como a aquisição e tratamento dos dados dos sensores, aumentando a produtividade e diminuindo o custo das aplicações desenvolvidas.

3.6.4 Capacidade de Monitoramento/Tratamento do Tipo Inscrição/Notificação

Existem duas formas básicas das aplicações *pervasivas* obterem dados de contexto: através do modelo inscrição/notificação, discutido na seção seguinte, ou através de consultas síncronas. O modelo de consultas síncronas é o mais utilizado nas aplicações convencionais. Quando o programador da aplicação deseja que uma condição atual do ambiente seja testada, o sistema efetua o cálculo do valor atual da condição que então é avaliada e a decisão é tomada. O fluxo decisório aguarda até que o status seja atualizado. No caso de variáveis de contexto cujos dados vem de sensores esse processo pode acrescentar um tempo significativo ao processo. Ainda assim, esse método é importante para programadores de aplicações *pervasivas* devido a sua popularidade em aplicações tradicionais.

3.6.5 Capacidade de Monitoramento/Tratamento sob Demanda

Enquanto no modelo tradicional de desenvolvimento de aplicações usa-se consultas síncronas, na Computação Pervasiva as aplicações devem reagir a modificações no ambiente. Neste caso, o programador da aplicação insere um gatilho que é ativado quando determinado evento acontece.

Nenhuma ação é executada até que aconteça o evento disparador do gatilho. Esse comportamento pode ser simulado através de consultas síncronas sucessivas, buscando detectar modificações no ambiente e acionando uma ação apenas quando o evento disparador ocorrer. O custo de efetuar repetidas consultas indefinidamente, entretanto, pode ser proibitivo, principalmente em um cenário onde muitas aplicações estarão em execução concorrentemente.

3.6.6 Tratamento Parametrizável, Adaptável a Cada Aplicação

Cada aplicação pode reagir diferentemente a um determinado estado do contexto, ou mesmo relacionar diferentemente os dados dos sensores gerando novas informações. O *middleware* de reconhecimento de contexto genérico deve ser capaz de personalizar o tratamento das informações de acordo com a necessidade descrita por cada aplicação, para tanto, deve especificar uma interface de comunicação para que os programadores de aplicação realizem essas descrições.

3.6.7 Capacidade de Composição Novos Dados de Contexto com Reaproveitamento de Tratadores

O contexto é percebido pelos sistemas computacionais através de sensores, que muitas vezes provêem informação de forma bruta, que precisa ser processada para se tornar útil às aplicações. Há diferentes técnicas para transformar informações de contexto gerando novos dados. Existem sistemas onde a cada tratamento deve ser programado independentemente. Em outros sistemas, por outro lado, o tratamento das informações é feito em módulos reaproveitáveis que podem ser combinados e parametrizados diferentemente, de acordo com as necessidades de cada situação. O reaproveitamento de filtros já existentes facilita o processo de desenvolvimento de aplicações, já que o programador de aplicações pode utilizar elementos já prontos e previamente testados.

3.6.8 Tabela Comparativa

A Figura 3.1 mostra quais das características listadas na seção anterior são atendidas pelos *middlewares* estudados neste capítulo. Pode-se perceber pelo quadro comparativo que nenhum deles consegue atender a todos os requisitos dispostos na seção 3.6. O Context Toolkit é focado em uma metodologia de desenvolvimento de aplicações conscientes do contexto, teve importância na motivação de trabalhos na área, mas mantém o tratamento dos dados do contexto como tarefa da própria aplicação. O projeto Solar apresenta uma proposta interessante para o compartilhamento de informações de contexto. Entretanto, não existe uma forma simples para a combinação e parametrização de filtros, chamado operadores. Além disso, oferece suporte limitado à desconexão, característica presente na computação móvel.

Tabela 3.1: Comparação entre principais sistemas para reconhecimento de contexto.

	Context Toolkit	Solar	Aura /CIS	JCAF	EgoSpaces
Compartilhamento dos dados monitorados	X	X	X	X	X
Capaz de integrar informações de diferentes sensores num mesmo dado tratado	X	X	X	X	-
Compartilhamento dos dados tratados	X	X	-	X	X
Independência entre tratamento e aplicação	-	X	X	-	-
Suporte à mobilidade / desconexão	-	-	X	-	X
Capacidade de monitoramento/tratamento sob demanda	X	X	X	X	X
Capacidade de monitoramento/tratamento do tipo inscrição/notificação	X	X	-	X	-
Tratamento parametrizável, adaptável a cada aplicação	-	-	X	X	-
Capaz de compor novos dados de contexto sem necessidade de programação	-	X	X	-	-

O tratamento dos dados de contexto é feito no projeto Aura/CIS em uma camada de *software* chamada *query sinthethizer* que simula o funcionamento de um SGBD, onde até mesmo a linguagem de consulta é próxima à SQL utilizada em sistemas de banco de dados. Todas as consultas são realizadas explicitamente, não havendo o mecanismo de inscrição/notificação, de grande importância em sistemas que pretendem reagir a modificações no ambiente.

O projeto JCAF apresenta um *framework* para o desenvolvimento de aplicações conscientes do contexto, o tratamento dos dados dos sensores é de responsabilidade de cada aplicação. Adicionalmente, todo tratamento precisa ser programado já que não existe nenhum mecanismo para compor diferentes filtros e gerar uma nova informação de contexto.

A bibliografia do projeto EgoSpace não menciona como a informação dos sensores deve ser monitorada ou como ela deve ser tratada. O projeto é focado no contexto sob o ponto de vista da aplicação de um único usuário e sua interação com os demais. Não

existe capacidade de integrar informações compondo novos dados de contexto ou de utilização do mecanismo de inscrição/notificação.

O projeto ISAM esboça o funcionamento de um servidor de contexto. Entretanto, a versão atual do mesmo não é capaz de integrar informações de diferentes sensores.

4 MULTIS: MULTI-SENSOR CONTEXT SERVER

A Computação *Pervasiva*, onde informações e recursos são trazidos do ambiente computacional ao mundo real é, atualmente, tema para diversos trabalhos científicos, motivados pelas possibilidades apresentadas por esse novo paradigma que se mostram extremamente promissoras. Parte importante desses esforços é a respeito da capacidade dos sistemas computacionais de detectar o estado corrente e as transformações que ocorrem no ambiente, e adaptarem-se de forma a interagir da melhor forma possível com o usuário.

Ao longo deste capítulo será detalhada a arquitetura do MultiS, um servidor de contexto voltado à Computação *Pervasiva*. Inicialmente se analisa a estrutura do processo de Reconhecimento do Contexto e identifica-se o foco do MultiS. Os elementos que compõem sua arquitetura são descritos na seção 4.2. Além do Servidor de Contexto, foi criada uma linguagem para a composição de sensores, que é apresentada na seção 4.3. Posteriormente, na seção 4.4, é detalhado o modulo responsável pelo processamento das informações do contexto, o `Context-Compositor`. A seção 4.5 discute o compartilhamento das informações de contexto entre diversas aplicações. A seção 4.6 apresenta os requisitos necessários ao funcionamento do MultiS e analisa sua interface com esses sistemas. Finalmente na seção 4.7 são resgatados os aspectos importantes da Computação *Pervasiva* e analisa-se como cada um desses aspectos é atendido pelo MultiS.

4.1 Morfologia do Processo de Reconhecimento de Contexto

O processo de consciência do contexto pode ser dividido em três etapas, conforme demonstra a Figura 4.1. Inicialmente, é necessário que se adquira informações do ambiente, tarefa realizada através de sensores. Cada um deles coleta dados a respeito de determinado aspecto do contexto, proporcionando ao sistema uma visão parcial da realidade. Esta área foi tema de diversos trabalhos, e maiores detalhes sobre os problemas envolvidos no processo de monitoração do ambiente podem ser encontrados em (SILVA, 2003).

No outro extremo da figura está a camada de adaptação. É de sua responsabilidade a efetivação da transformação do comportamento das aplicações em caso de modificações do ambiente. Na camada intermediária, tema deste trabalho, ocorre o reconhecimento de contexto. Os dados dos sensores são combinados de forma a gerar outras informações que são utilizadas no processo de adaptação.

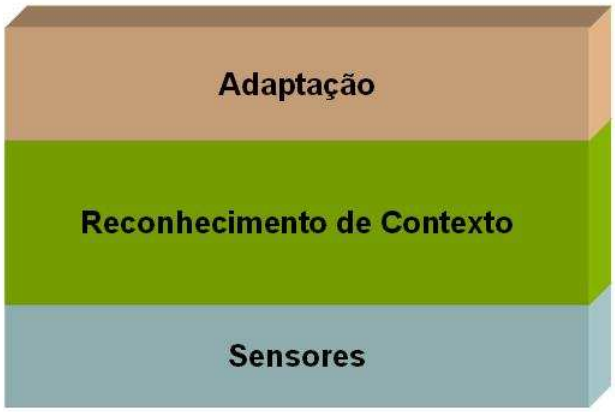

Figura 4.1: Etapas envolvidas no processo de consciência do contexto.

Os principais trabalhos que se propõem a resolver o problema do reconhecimento de contexto foram estudados no Capítulo 3, dentre eles destacam-se o Solar, o Context Toolkit e o Aura/CIS. O Solar apresenta o conceito de grafo de operadores no tratamento de informações, que são disponibilizadas através de fluxos de dados. Os operadores estão espalhados entre os diversos computadores do sistema. Na Computação *Pervasiva*, no entanto, pela própria natureza móvel dos dispositivos, a desconexão é uma realidade sempre presente e pode resultar em remoção ou inserção de nodos em tempo de execução. A desconexão de um nó que contenha um operador, nesta arquitetura, pode prejudicar o processo de consciência do contexto.

O Context-Toolkit defende uma metodologia de desenvolvimento de aplicações conscientes do contexto e apresenta uma arquitetura para sua execução. Nela, os tratadores de informações fazem parte de cada aplicação, obrigando o programador a sempre desenvolver o filtro que for necessário à aplicação que está desenvolvendo. Também não são tratadas questões relativas à mobilidade da aplicação. O projeto Aura utiliza uma camada que abstrai o processo de consulta a informações do contexto, simulando um sistema de consultas de um banco de dados. É uma solução interessante para calcular o valor da integração dos sensores. Entretanto, esse método não oferece um mecanismo de reatividade a modificações do ambiente, obrigando as aplicações a realizarem consultas periódicas para reagirem a modificações do contexto.

Dentre os trabalhos estudados, nenhum conseguiu atender aos requisitos impostos pela Computação *Pervasiva*. Esses requisitos, que nortearam o desenvolvimento deste trabalho, encontram-se brevemente apresentado abaixo.

- Facilidade de desenvolvimento;
- Suporte à mobilidade e desconexão;
- Flexibilidade e extensibilidade;
- Adaptabilidade e reatividade ao ambiente;

O MultiS, detalhado neste capítulo, apresenta uma proposta para um servidor de contexto voltado às aplicações *pervasivas* conscientes do contexto. Objetiva, portanto, atender aos requisitos apresentados acima.

4.2 Seqüência de Eventos em um Caso de Uso

O MultiS é um sistema de reconhecimento de contexto direcionado às aplicações *pervasivas*. A função de um servidor de contexto é relacionar as informações coletadas do ambiente possibilitando a adaptação das aplicações. O MultiS desempenha as funções da camada intermediária da Figura 4.1, apresentada na seção anterior. Serviços externos de monitoração e adaptação devem ser utilizados em conjunto com o MultiS. O caso de uso (BOOCH, 1999) representado na Figura 4.2 ilustra o fluxo de eventos no sistema segundo o modelo proposto.

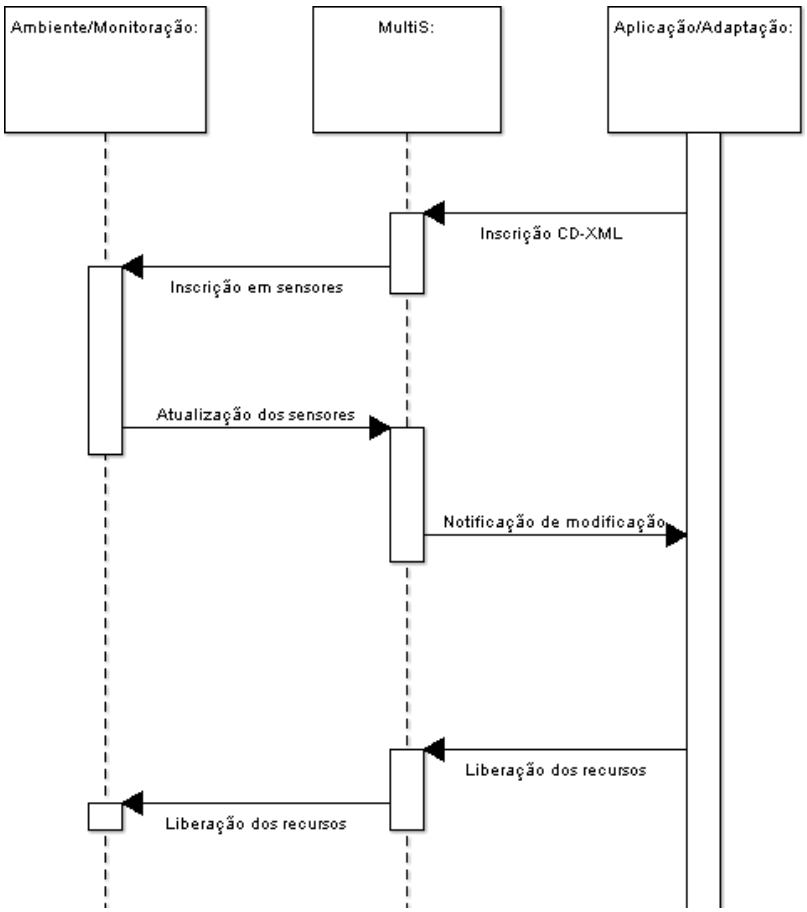

Figura 4.2: Diagrama de seqüência com o ciclo de vida tradicional de uma aplicação utilizando MultiS.

Uma aplicação submete uma requisição ao MultiS descrevendo seu contexto de interesse. As requisições de inscrição são feitas na linguagem CD-XML, criada especialmente para esse fim, e descrevem como os dados dos sensores disponíveis no ambiente devem ser relacionados para a geração de dados de contexto de alto nível, aos quais a aplicação será sensível. A linguagem CD-XML é detalhada na seção 4.4.

O MultiS inscreve-se, então, no sistema de monitoramento, solicitando que as informações relativas aos sensores necessários ao cumprimento da requisição lhe sejam transmitidas. Qualquer modificação no valor de algum dos sensores dispara um recálculo no sistema de reconhecimento de contexto e que pode resultar em adaptação da aplicação.

Eventualmente, uma modificação no ambiente é detectada pelo sistema de monitoramento que notifica o sistema de reconhecimento de contexto. O novo estado do ambiente é avaliado, caso seja necessário, o sistema de adaptação é ativado para que a aplicação tenha seu comportamento modificado. Esse passo pode repetir-se indefinidamente, com sucessivas transformações no ambiente e no comportamento da aplicação.

Ao término do seu funcionamento, a aplicação notifica o sistema de reconhecimento de contexto, que pode liberar recursos utilizados no cumprimento da requisição. Essa informação é repassada ao sistema de monitoramento que pode agir de forma similar.

4.3 Componentes da Arquitetura

A arquitetura MultiS é composta por diversas camadas com funções distintas. A Figura 4.3 apresenta os componentes da arquitetura. Buscando facilitar o entendimento, a figura também contém a representação dos sistemas de monitoração e de adaptação, que não fazem parte da arquitetura do sistema, mas são requeridos para seu funcionamento.

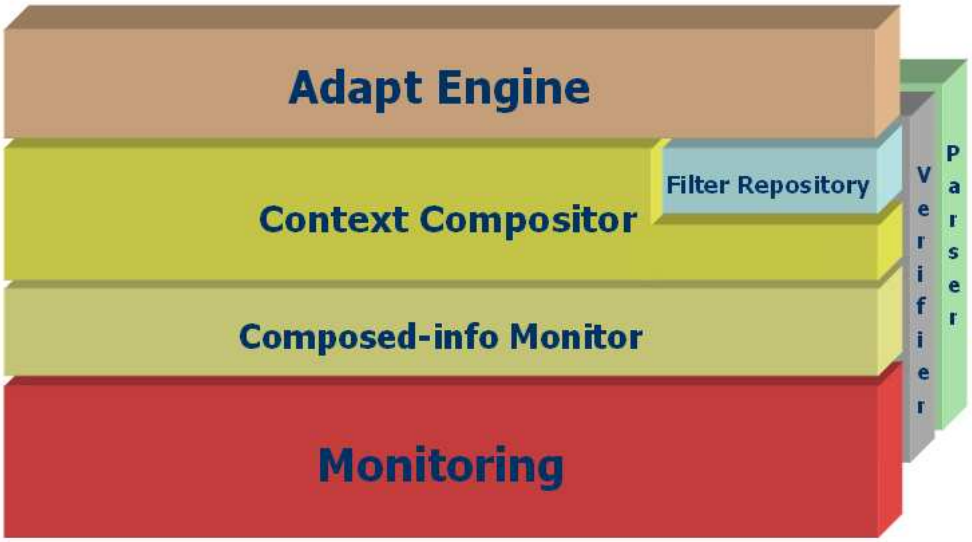

Figura 4.3: Arquitetura MultiS.

4.3.1 `Parser`

O módulo `Parser` é responsável por receber a definição de contexto das aplicações e repassá-la ao interpretador da linguagem utilizada na requisição. Após o processamento da requisição, as informações obtidas são repassadas ao módulo `Verifier`, responsável pela alocação dos recursos necessários para que a requisição seja cumprida.

4.3.2 `Verifier`

O módulo `Verifier` expõe uma API que é utilizada pelo módulo `Parser`. Ela contém funcionalidades que permitem a carga dinâmica de código de execução, além da alocação e desalocação de recursos nas demais partes do sistema de reconhecimento do

contexto. Durante esse processo, busca-se a reutilização de elementos já existentes no ambiente. Caso seja necessário, novos sensores podem ser solicitados à camada de monitoração. Se ainda assim a requisição não puder ser cumprida, a aplicação é notificada com uma exceção.

4.3.3 `Monitoring`

Os sensores são elementos essenciais da Computação Consciente do Contexto. É através deles que o sistema percebe o ambiente que o cerca. Existem diferentes técnicas para aquisição de informações e diversos estudos já foram realizados sobre isso. As peculiaridades de cada uma das estratégias impedem a formulação de um sistema genérico capaz de lidar com os diversos tipos de sensores existentes. Por esse motivo, grande parte dos trabalhos que tratam do reconhecimento de contexto utilizam informações de serviços especializados em monitoração.

O MultiS também adota essa estratégia, utilizando uma camada externa de sensoriamento de informações. Sugere-se, para uso conjunto com o MultiS, a utilização de um *middleware* de sensoriamento de informações capaz de reunir informações provenientes de diversas fontes e disponibilizá-las através de uma interface única, um exemplo de sistema de monitoramento desse tipo é o PRIMOS (SILVA, 2003), que é o subsistema responsável pelo monitoramento na arquitetura ISAM.

Destaca-se que o sistema de monitoramento pode não ser capaz de determinar o valor de um dado sensor (*eg.* em casos de desconexão), admite-se, portanto, o uso do valor indeterminado para o sensor, situação esta que pode ser tratada pelo MultiS. É importante que o Serviço de Monitoração seja capaz também, de comunicar ao sistema de reconhecimento de contexto quando ocorrerem modificações nos valores dos sensores que estão sendo utilizados.

4.3.4 `Context-Compositor`

O `Context-Compositor` é o núcleo do sistema de reconhecimento de contexto. Ele é responsável pela integração e processamento dos dados dos sensores. Esse processamento é realizado utilizando-se filtros organizados em árvores de inferência de contexto, cujo funcionamento é detalhado na seção 4.5. Caso as situações determinadas pela aplicação em sua especificação de contexto sejam atendidas, os dados produzidos pelo `Context-Compositor` são direcionados à camada de adaptação.

4.3.5 `Filter Repository`

Os filtros são bibliotecas utilizadas para o processamento dos dados de contexto. Eles são armazenados em uma base chamada repositório de filtros. Sua carga é feita dinamicamente pelo módulo `Verifier`, eles são utilizados no `Context-Compositor` para a produção de informações de contexto.

O `Filter Repository` é um repositório de filtros e contém informações sobre os filtros existentes, permitindo que os programadores escolham os filtros que serão utilizados pelas aplicações. A adição de novos filtros no sistema acontece através da inserção de elementos no repositório. Essa operação é efetuada pelo administrador do repositório e pode ser realizada a qualquer tempo, sem que seja necessário reiniciar o Serviço de Reconhecimento de Contexto.

4.3.6 `Composed-Info Cache`

A `Composed-Info Cache` (CIC) é uma camada de abstração de acesso aos dados de contexto. Sua função é interagir com a camada de monitoração e com o módulo `Context-Compositor` para disponibilizar informações de contexto. As informações são consultadas em duas fontes: o *cache* interno do CIC, que contém informações de contexto previamente produzidas pelo `Context-Compositor`, ou através de consulta ao Serviço de Monitoração.

4.3.7 `Adapt-Engine`

O `Adapt-Engine` é o serviço externo ao servidor de contexto que tem a função de selecionar o novo comportamento da aplicação quando ocorre uma modificação no contexto. Esses eventos de adaptação chegam à camada de adaptação através do módulo `Context-Compositor`. Destaca-se que a modificação em um único dado do contexto pode refletir em diversos eventos de adaptação.

4.4 Linguagem para Composição de Sensores

No modelo de tratamento de contexto proposto neste trabalho, as informações são processadas no servidor de contexto, que dispara um evento de adaptação em um sistema externo, quando ocorre uma modificação no ambiente. O programador de cada aplicação deve descrever os aspectos do ambiente os quais uma mudança deve resultar em modificação no comportamento da aplicação. Esta descrição é utilizada pelo servidor de contexto para o processamento dos dados dos sensores e alerta ao sistema de adaptação.

É necessário, portanto, que exista uma maneira com a qual as aplicações consigam descrever como as informações dos sensores devem ser processadas e em que situações o programador deseja que a aplicação seja notificada da mudança de um determinado dado. Diversos sistemas de reconhecimento existentes propõem a utilização de linguagens específicas para esse fim.

O projeto Aura/CIS defende a utilização de uma extensão da linguagem SQL, que por ser baseada em uma linguagem amplamente conhecida, é de simples aprendizado. Sua utilização é indicada para consultas explícitas e síncronas, mas não é, no entanto, expressiva o suficiente para comportar o modelo inscrição/notificação utilizado na Computação Consciente do Contexto. Os autores do projeto Solar defendem a utilização de uma linguagem baseada em XML que descreve como os operadores se relacionam, a idéia entretanto, não foi aprofundada por esse grupo. Não existe na literatura estudada, nenhuma solução satisfatória para esse problema.

Diversos são os desafios enfrentados na elaboração de tal proposta. Novos sensores podem ser acrescentados e novos filtros podem ser disponibilizados durante a execução do servidor de contexto. A linguagem de descrição do contexto utilizada entre o servidor de contexto e as aplicações precisa ser, portanto, facilmente extensível, comportando a adição desses novos elementos. Ainda se faz necessário o suporte às características da Computação *Pervasiva*, onde nodos móveis podem ficar indisponíveis a qualquer momento, possivelmente resultando no estado indefinido dos sensores neles localizados. Além disso, a linguagem deve ser simples e facilmente utilizável pelos programadores de aplicação. A seguir é apresentada a linguagem CD-XML, criada com o objetivo de atender a esses requisitos.

4.4.1 A Linguagem CD-XML

A linguagem CD-XML (*Context-Definition* XML) foi criada para que as aplicações conscientes do contexto possam descrever como devem ser relacionados os dados dos sensores para produzir informações de contexto de alto nível, permitindo que as aplicações reajam a modificações no ambiente e adaptem seu comportamento.

Cada trecho da linguagem relaciona informações de fontes de dados através de filtros, que processam os dados produzindo novas informações. Os dados de entrada utilizados nos filtros podem ser provenientes da camada de monitoramento ou informações que já foram previamente processadas por outro trecho da linguagem. A informação resultante do processamento do filtro pode ser utilizada como fonte de dados para outro filtro, e assim sucessivamente.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="context">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="filter"/>
                <xsd:choice maxOccurs="unbounded" minOccurs="1">
                    <xsd:element ref="context"/>
                    <xsd:element ref="sensor"/>
                </xsd:choice>
            </xsd:sequence>
            <xsd:attribute name="label" type="xsd:string"
use="required"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="filter">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element ref="parameters"/>
            </xsd:sequence>
            <xsd:attribute name="name" type="xsd:string"
use="required"/>
            <xsd:attribute name="default" type="xsd:string"
use="optional"/>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="parameters">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element maxOccurs="unbounded" minOccurs="0"
ref="parameter"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="parameter" type="xsd:string"/>
    <xsd:element name="sensor" type="xsd:string"/>
</xsd:schema>
```

Figura 4.4: XML Schema com a sintaxe da linguagem CD-XML.

Quando uma informação de contexto mais externa de uma dada requisição é modificada, o sistema de monitoramento é ativado. Ele recebe os dados a respeito do novo estado do contexto e pode ativar o processo de adaptação da aplicação. A Figura

4.4 apresenta o XML Schema que define a linguagem CD-XML. A seguir, cada um dos tipos de elementos presentes na linguagem é detalhado.

- **`<sensor>:`** Representa um sensor disponível no sistema de monitoramento. O valor de uma etiqueta `<sensor>` (*eg.* `isComputerInUse`) é utilizado para a identificação do sensor no sistema de monitoramento.

- **`<context>:`** Representa uma informação que é processada pelo sistema de reconhecimento de contexto. As informações contidas nesse elemento indicam os dados que são utilizados para produzir esse valor. Cada etiqueta deste tipo possui um filtro e pelo menos um provedor de informações. O filtro é indicado pela etiqueta `<filter>` enquanto os provedores de informação podem ser etiquetas do tipo `<sensor>`, quando forem utilizados dados diretamente do sistema de monitoramento, ou outros elementos `<context>`, neste caso, a informação é produzida pelo próprio sistema de reconhecimento de contexto.

- **`<filter>:`** Representa um filtro utilizado para processar de informações de contexto. A propriedade `name` de um elemento `<filter>` é utilizada para a identificação de qual filtro será utilizado no processamento das informações. Adicionalmente, um filtro pode receber parâmetros que modificam o comportamento do filtro, representados pela etiqueta `<parameters>`. Outra propriedade importante dos filtros é determinada pela propriedade `default` que é opcional, e pode ser utilizada para determinar o valor retornado por um filtro quando não for possível obter o valor de algum sensor. Maiores detalhes na seção 4.4.2

- **`<parameters> e <parameter>:`** Contém informações relativas à parametrização de um filtro. A entrada `<parameters>` é um invólucro para diversos parâmetros, cada um especificado em uma etiqueta `<parameter>.`

4.4.2 Tratamento de Valores Indeterminados

Uma das principais características da Computação Móvel é que os nodos móveis são muito suscetíveis a defeitos de comunicação e a desconexão. Tendo em vista que alguns sensores utilizados pelo servidor de contexto podem estar localizados em dispositivos móveis, é necessário que exista um mecanismo para produzir valores corretos para os dados de contexto quando essas situações ocorrerem.

O comportamento do filtro nos casos onde os valores de entrada são indeterminados pode ser previsto e especificado pelo programador. Muitos filtros, entretanto, são genéricos e utilizados em inúmeras situações diferentes. Essas situações podem requerer diferentes comportamentos do mesmo filtro para o caso de entrada de valores indeterminados.

Suponha-se, por exemplo, que o filtro que agrega os valores de diversas fontes em uma única lista receba de uma das fontes a informação que seu valor não pode ser determinado no momento. O comportamento do filtro nesse caso pode depender da situação em que ele está inserido. Se a situação em que o filtro está sendo utilizado requerer que absolutamente todos os sensores apresentem um valor válido, o valor retornado deverá ser indeterminado. Por outro lado, se a utilização requerer todos os

valores possíveis, mas admite que algum sensor possa ser indeterminado, o mesmo filtro também poderá se comportar dessa forma.

Para modificar o comportamento de um filtro em caso de valores indeterminados, deve-se informar o valor padrão do filtro através da propriedade `default` da etiqueta `<filter>`. Essa cláusula deve ser usada apenas nas situações onde se deseja que determinado valor seja produzido pelo filtro ao invés do valor indeterminado.

4.4.3 CD-XML: Exemplo de Utilização

Com o propósito de demonstrar um caso real de utilização da linguagem CD-XML, apresenta-se o trecho CD-XML da Figura 4.5. O código é utilizado por uma aplicação que determina a presença de algum usuário em uma sala em horário comercial e notifica essa presença a uma aplicação, permitindo que ações previamente configuradas sejam disparadas automaticamente.

```
<?xml version="1.0" encoding="UTF-8"?>
<context label="WhenInRoom"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="cd-xml.xsd">
            <filter name="alert">
                  <parameters>
                        <parameter>present</parameter>
                  </parameters>
            </filter>
        <sensor>currentTime</sensor>
      <context label="isPersonInRoom">
            <filter name="FilterAtLeast" default="false">
                  <parameters>
                        <parameter>3</parameter>
                  </parameters>
            </filter>
            <sensor>isComputerInUse</sensor>
            <sensor>isChairInUse</sensor>
            <sensor>isMovActive</sensor>
            <sensor>isLightOn</sensor>
            <sensor>isDoorLocked</sensor>
      </context>
</context>
```

Figura 4.5: Exemplo de utilização da linguagem CD-XML.

Na Tabela 4.1 a seguir, são detalhados os componentes (sensores e filtros) utilizados pela aplicação.

Tabela 4.1: Componentes utilizados na aplicação de detecção de presença.

Tipo	Nome	Descrição
Sensor	`isComputerInUse`	Indica se o computador está sendo utilizado por um usuário. A informação é positiva se o computador estiver ligado e o protetor de tela não estiver ativo.

Sensor	`IsChairInUse`	Indica se a cadeira principal está sendo utilizada. A informação é positiva caso exista algum peso sobre o assento.
Sensor	`IsMovActive`	Utilizando um sensor infravermelho de movimento determina se existe movimentação na sala.
Sensor	`IsLightOn`	Indica se a luz da sala estáe acesa
Sensor	`IsDoorLocked`	Indica se a porta da sala encontra-se trancada.
Sensor	`currentTime`	Indica o horário
Filtro	`FilterAtLeast`	Este filtro recebe um parâmetro que indica quantos provedores de informação são necessários para que o resultado do filtro seja positivo.

O objetivo da aplicação é reagir à presença de pessoas em uma sala, para tanto, são utilizados seis sensores disponíveis no ambiente. Cada sensor isoladamente não é suficiente para resolver o problema de determinar a presença de alguém na sala. O sensor `isChairInUse`, por exemplo, pode fornecer informações incorretas se um objeto for deixado sobre o assento da cadeira. Mas ao integrar essa informação com outros sensores, diminui-se a chance de falsos positivos, isto é, informações incorretas de apenas um dos sensores são suprimidas pelos outros sensores, preservando a correção do comportamento da aplicação.

4.4.4 Desenvolvendo em CD-XML

Existem atualmente diversas ferramentas especializadas em edição de documentos XML. Quando configuradas adequadamente e utilizando o arquivo com o XML Schema que define as regras de formação da linguagem CD-XML, pode-se aproveitar as facilidade oferecidas por ferramentas já existentes para programar utilizando, até mesmo, uma interface visual. A verificação da correção da descrição de contexto CD-XML é feita em tempo de programação, agilizando o desenvolvimento e a correção de defeitos na sintaxe da programação. A Figura 4.6 mostra a edição visual de um arquivo CD-XML na ferramenta WebSphere Studio Application Developer [WSAD], da IBM.

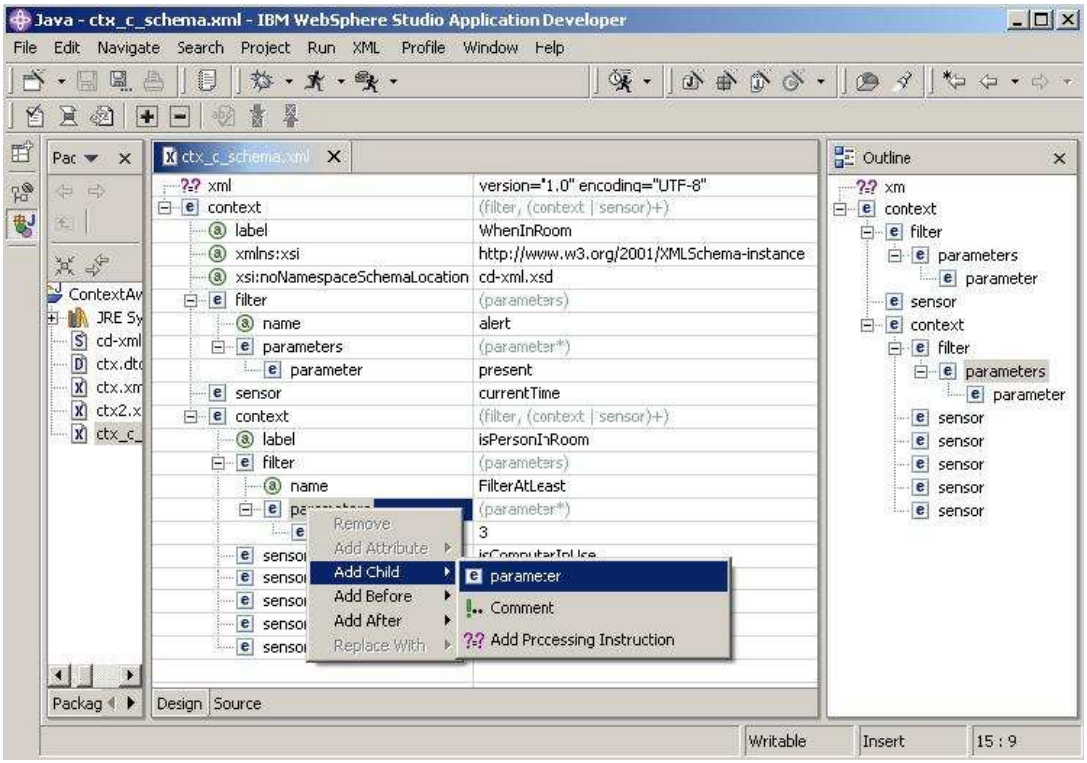

Figura 4.6: Edição visual de código CD-XML na ferramenta WebSphere Studio Application Developer.

É importante destacar que mesmo com a utilização desse tipo de ferramenta, o programador ainda precisa ter conhecimento de quais sensores estarão disponíveis no momento em que a aplicação for executada e quais filtros estarão disponíveis no repositório. A identificação dos sensores e dos filtros é realizada através de um nome único que deve ser conhecido do programador. Atualmente não são oferecidos na linguagem mecanismos de procura de sensores. Sugere-se a extensão da linguagem para selecionar sensores em tempo de execução de acordo com suas características como trabalho futuro.

4.5 Mecanismo de Inferência de Informações de Contexto

Esta seção aborda o funcionamento do mecanismo de inferência de informações do contexto. Inicialmente discute-se a detecção de alterações no ambiente e o processamento dessas informações pelo sistema de reconhecimento do contexto, e em seguida é apresenta uma proposta para o compartilhamento de informações de contexto objetivando a economia de recursos do *middleware*.

4.5.1 Percebendo Modificações nos Sensores

Os sensores são os nodos que recebem as informações do ambiente, e é a alteração dos seus valores que inicia processos de atualização das variáveis de contexto, que pode resultar em modificação no comportamento da aplicação. O cálculo é iniciado quando um sensor tem seu valor modificado.

O recebimento de uma notificação de modificação em uma variável de contexto resulta no recálculo das variáveis de contexto que utilizam a informação modificada. As

informações são avaliadas de acordo com os valores dos sensores no momento em que a notificação foi recebida. Caso novas modificações ocorram no ambiente durante o processo de recálculo efetuado pelo MultiS, os novos valores dos sensores são armazenados em memória. Esses valores serão percebidos apenas no ciclo seguinte de modificações de variáveis de contexto, quando todos os eventos recebidos podem então ser avaliados. Essa fila de eventos tem um tamanho limitado e caso seu tamanho máximo seja atingido por um grande número de eventos, algumas modificações no ambiente podem ser perdidas.

4.5.2 Árvores de Inferência do Contexto

Quando uma modificação no ambiente é detectada, o sistema de reconhecimento de contexto é ativado. Segundo a proposta do MultiS, as informações dos sensores são relacionadas através de uma estrutura de dados em forma de árvore previamente existente no servidor de contexto. Essas árvores são criadas no momento em que a aplicação inscreve-se no sistema, e são obtidas através do processamento das requisições CD-XML. Na Figura 4.7, apresenta-se de forma esquemática a estrutura gerada a partir do processamento do trecho CD-XML apresentado na Figura 4.5.

Os elementos `<context>` sempre representam o nodo raiz de uma árvore ou sub-árvore de inferência. Associado ao nodo raiz, sempre existe um filtro e pelo menos um provedor de informação, que pode ser um elemento do tipo <sensor> ou outro nodo `<context>`. O valor do nodo `<context>` é determinado pela execução do filtro, dado o estado dos provedores de informação associados.

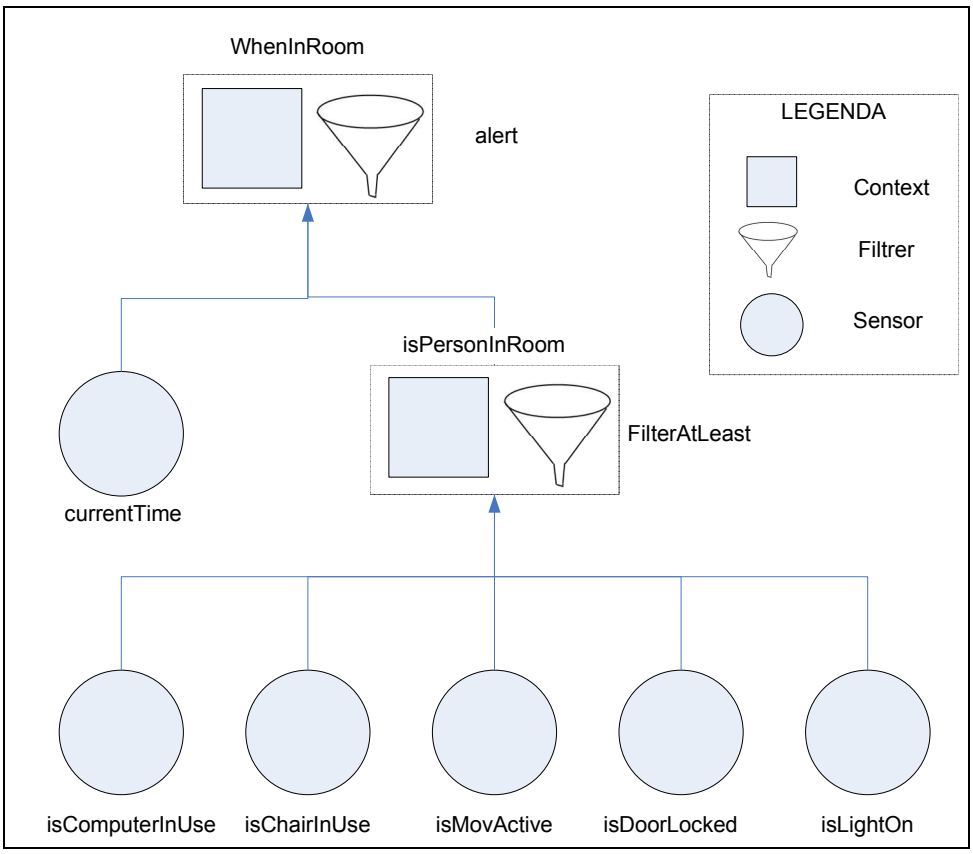

Figura 4.7: Árvore de inferência de contexto.

Por exemplo, para obter o valor do elemento `<context> isPersonInRoom` do trecho CD-XML da Figura 4.5, o sistema invoca o filtro `FilterAtLeast`, que recebe informação disponibilizada pelos provedores de informação diretamente abaixo dele: `isComputerInUse`, `isChairInUse`, `isMovActive`, `isDoorLocked`, `isLightOn`. Modificações nos dados fornecidos por esses sensores podem gerar um evento de adaptação conforme detalhado na seção seguinte.

Destaca-se que todos os nodos folha das árvores de inferência são do tipo `sensor`, e todos os nodos `sensor` são folhas. Como qualquer modificação do ambiente que dispara um recálculo do valor da árvore é iniciada por um `sensor`, as informações são sempre processadas de maneira *bottom-up*.

4.5.3 Processamento do Valor das Árvores de Inferência de Contexto

Se o cálculo do valor de uma sub-árvore implicar a alteração do valor do elemento raiz, o elemento que utiliza essa informação também precisará ser recalculado, e assim sucessivamente. Eventualmente essas modificações podem gerar uma notificação ao sistema de adaptação resultando em transformação do comportamento da aplicação.

Destaca-se que nem todas as modificações em sensores geram eventos de adaptação. Se a alteração em um provedor de informação for insuficiente para modificar o valor de um elemento, não será necessário prosseguir com os recálculos das árvores que utilizam a informação do nodo.

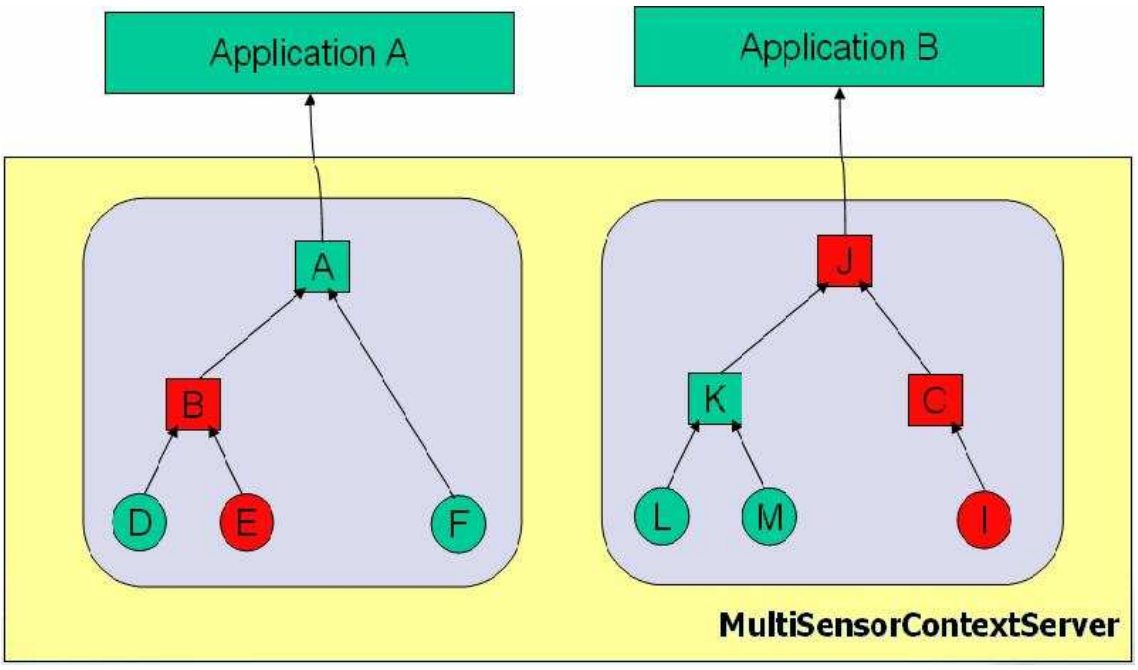

Figura 4.8: Propagação de modificações em variáveis de contexto após modificação no ambiente.

O recálculo do valor representado por uma sub-árvore é dado pela execução do filtro associado ao elemento `context` que representa esta sub-árvore. Para produzir esse

valor, o filtro utiliza o valor de todos os provedores de informação associados ao elemento `context`, sejam eles sensores ou outros elementos `context`. A implementação desse algoritmo é simples com a utilização de recursividade. A Figura 4.8 demonstra o comportamento descrito em duas situações, que são detalhadas a seguir.

Uma verificação junto ao sistema de monitoramento detectou que dois sensores, representados pelos nodos E e I na figura, tiveram seus valores atualizados. O sistema deve, portanto, verificar se estas modificações alteram significativamente o ambiente e, portanto, devem gerar eventos de adaptação. A modificação no estado do sensor E dispara um evento de recálculo do valor representado pelo nodo B. Esse também tem seu valor modificado, e, portanto, o nodo A, que utiliza informação disponibilizada pelo nodo B também deve ser recalculado. Entretanto, a alteração de valor em B não é suficiente para alterar o valor do nodo A, e nenhum evento de adaptação precisa ser gerado. A alteração no ambiente que é detectada pelo sensor I, por outro lado, gera sucessivos recálculos em árvores de inferência de contexto até que a aplicação seja notificada. Neste exemplo não foi necessário recalcular o valor do nodo K, já que nenhum dos dados utilizados por ele foi alterado.

4.5.4 Filtros e seu Repositório

Os filtros são as unidades básicas de processamento de informações no modelo MultS, pois são os eles que têm a função de gerar informações de contexto a partir de outros dados. Também são responsáveis por conferir flexibilidade ao sistema, já que através da adição de novos filtros é possível combinar informações diferentemente e até de tratar novos tipos de sensores. A carga de novos filtros pode ser feita dinamicamente, permitindo a adição de elementos em tempo de execução.

Na fase inicial do desenvolvimento deste trabalho, planejou-se a criação de um conjunto limitado de operações para a implementação de filtros. O servidor de contexto receberia apenas a descrição de como relacionar os dados de contexto em CD-XML e utilizaria os filtros existentes para a realização desses cálculos. Esta solução é relativamente simples de ser implementada, mas seria muito difícil prever todos os filtros que seriam necessários para atender às diferentes situações que podem existir em um tipo ambiente tão variado quanto o da Computação Consciente do Contexto.

A alternativa escolhida foi a utilização de filtros programados, que podem conter qualquer trecho de código, desenvolvidos em uma linguagem de programação comum, como Java, e carregados em tempo de execução de acordo com a necessidade. Essa solução não tem a mesma limitação quanto à expressividade, já que os filtros podem conter qualquer tipo de código, mas percebeu-se um possível problema de segurança. Com a carga dinâmica de código novo e possivelmente não verificado, seria possível a execução de programas com defeitos de programação ou mesmo com código malicioso.

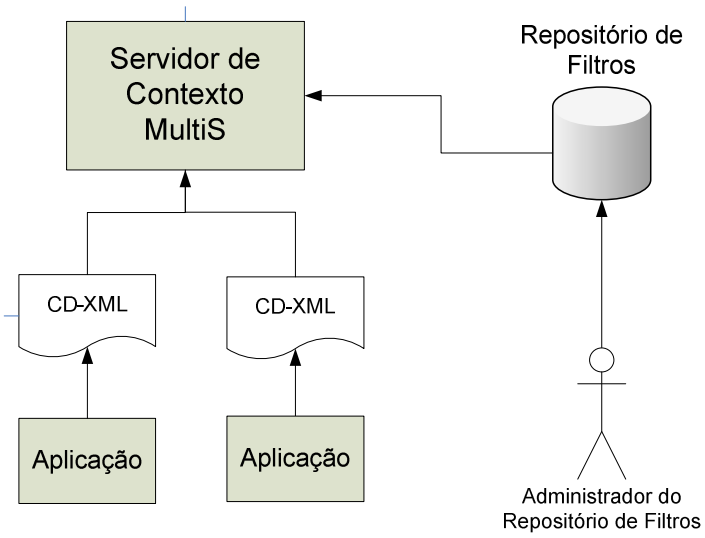

Figura 4.9: Repositório de Filtros.

Para resolver este problema, foi imposta uma restrição ao sistema. Só podem ser utilizados no servidor de contexto filtros que estejam armazenados em um Repositório de Filtros (`Filter Repository`). Ele contém apenas filtros que foram verificados por um Administrador de Filtros. O Administrador de Filtros é um usuário com privilégios administrativos sobre o Repositório, e apenas ele pode adicionar ou remover filtros do Repositório. Sua responsabilidade é garantir que novos filtros sejam verificados antes de serem disponibilizados, garantindo que eles não possuam código malicioso. Além disso, o Administrador deve garantir que uma descrição apropriada do filtro seja fornecida e publicada em um repositório chamado Catálogo de Filtros. O uso do Repositório é esquematizado na Figura 4.9.

O Catálogo contém uma descrição sobre o funcionamento dos filtros e quais são os pré-requisitos existentes para sua utilização. Informa, além disso, que tipo de informação de contexto é produzida pelo filtro e quais os parâmetros aceitos. Essa informação pode ser consultada pelos programadores de aplicação, permitindo-lhes que escolham o filtro apropriado para que a informação de contexto desejada seja produzida. A Tabela 5.2 é um exemplo de entradas no Catálogo de Filtros.

4.5.5 Compartilhando Informações Processadas

O Servidor de Contexto está apto a suportar diversas aplicações executando concorrentemente. As aplicações são inscritas em determinado servidor que provê informações a respeito do ambiente de interesse da aplicação (*eg.* uma sala, uma casa, etc.). Por esse motivo, as aplicações inscritas em dado servidor tendem a utilizar os mesmos sensores e algumas vezes, realizar o mesmo tratamento sobre esses sensores para obter dados de contexto (*eg.* a localização de determinado usuário, a detecção de atividade no ambiente, ou mesmo a identificação de quem está presente).

A Figura 4.10 representa a execução das árvores de inferência de contexto de três aplicações. É possível perceber que as Aplicações X e Y, embora tenham diferentes requisitos para o servidor de contexto, necessitam de uma informação comum, representada pela sub-árvore com raiz no nodo C.

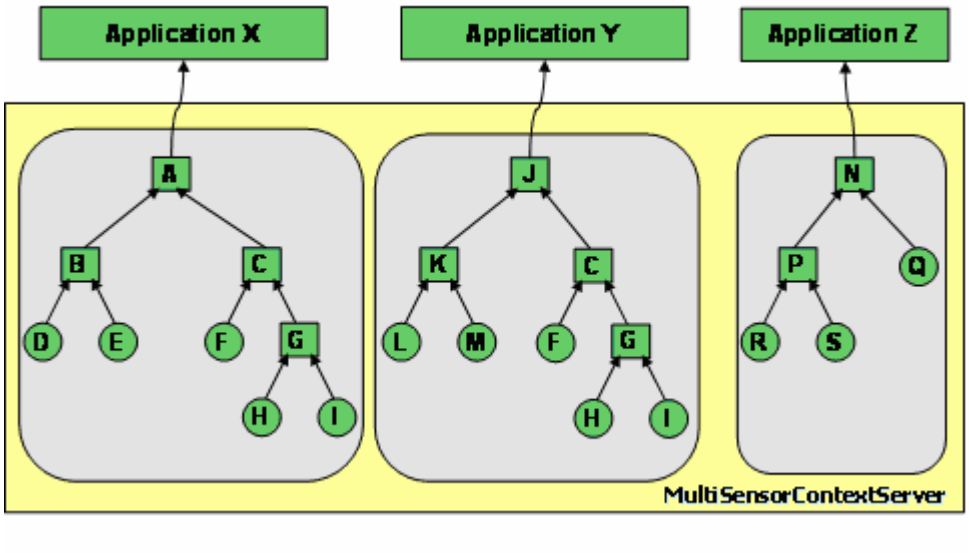

Figura 4.10 Diferentes aplicações em execução com sub-árvores equivalentes.

Existe, obviamente, um custo para processar cada uma das árvores de inferência de contexto. O MultiS busca economizar recursos do servidor suprimindo árvores de detecção de contexto sempre que possível. A supressão da árvore de contexto é realizada durante o processo de alocação de recursos para uma árvore. Antes de alocar cada uma das sub-árvores que descrevem o tratamento do contexto para aquela aplicação, o sistema verifica se já não existe uma outra sub-árvore que disponibilize a mesma informação. Caso essa existência seja confirmada, utilizam-se os recursos já alocados para obtenção da informação.

A identificação de árvores existentes é realizada através da busca de nodos de contexto equivalentes, ou seja, com a mesma descrição CD-XML. São consideradas equivalentes, apenas sub-árvores com a mesma descrição do contexto. Esse método de detecção foi escolhido por sua simplicidade, mas é importante mencionar que ele não detecta todas as sub-árvores possíveis de serem compartilhadas. Se duas aplicações apresentam descrições diferentes para o mesmo dado, apenas com diferenças sintáticas em sua descrição, o sistema não conseguirá identificar a semelhança e duas árvores distintas serão utilizadas. Um algoritmo de detecção de sub-árvores mais eficaz poderá resolver esse problema no futuro.

A supressão de uma árvore de inferência é ilustrada na Figura 4.11. A sub-árvore com raiz no nodo C, que na Figura 4.10 é repetida, é suprimida e o nodo raiz C, recebe dois apontadores que indicam os elementos que utilizam suas informações, os nodos A e J. Com esta modificação os elementos dispostos no servidor de contexto passam a ser representados internamente por grafos, e não árvores. Esta modificação, embora transparente para as aplicações, afeta a forma como as informações são processadas internamente pelo servidor de contexto e o algoritmo de cálculo do valor de árvores de inferência deve ser readequado.

Como visto anteriormente, o algoritmo de cálculo de informações de contexto pode ser implementado com a utilização de recursão. Com uma pequena modificação pode-se estender esse mesmo método para calcular o valor de grafos de inferência. Com a existência dos grafos de inferência, cada nodo pode ter apontadores para diversos outros

nodos, que são os consumidores de suas informações. O algoritmo pode ser modificado de tal forma que a alteração no valor de um nodo gere eventos de recálculo em todos os nodos que utilizam a informação do nodo alterado. Se o valor de C for alterado, por exemplo, as árvores representadas pelos nodos A e J precisarão ter seu valor atualizado. Esse novo algoritmo funciona apenas se o grafo não possui ciclos, o que sempre é verdade e é demonstrado no APÊNDICE A.

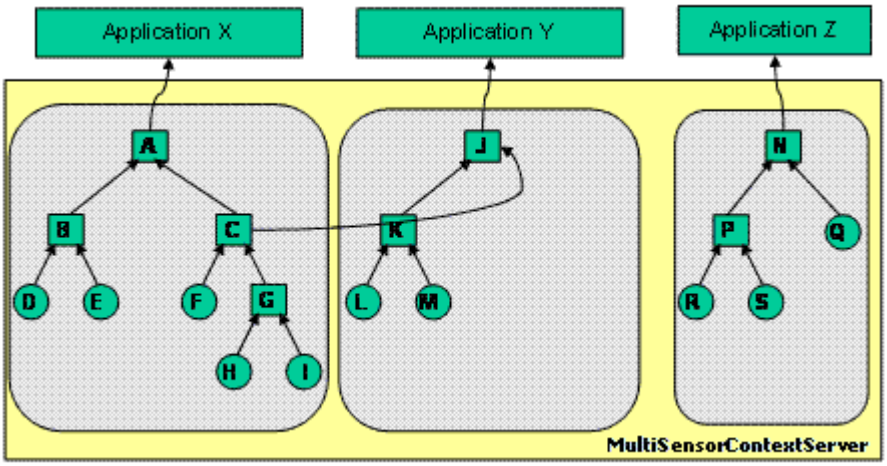

Figura 4.11: Reutilização de informações de contexto.

Do ponto de vista do programador da aplicação, o compartilhamento de informações é transparente. Cada aplicação recebe informações conforme requisitado na descrição de contexto que foi submetida ao servidor, não tendo ciência de que estas árvores podem estar funcionando como grafos dentro do servidor de contexto.

4.6 Requisitos para Funcionamento

Conforme discutido no início do capítulo, o processo de consciência do contexto pode ser dividido em três camadas com funções distintas: monitoramento, de reconhecimento do contexto e de adaptação. O MultiS, responsável pela camada intermediária, precisa ser associado a sistemas de monitoramento e adaptação para seu funcionamento. Essa sessão aborda a integração desses sistemas com o MultiS e quais requisitos precisam ser atendidos por esses sistemas para o pleno funcionamento do processo de reconhecimento de contexto.

4.6.1 Sistema de Monitoramento

A principal função do sistema de monitoramento é coletar dados do ambiente e disponibilizá-los ao sistema de reconhecimento do contexto. Num ambiente de Computação *Pervasiva*, os sensores podem estar dispersos em diversos dispositivos do ambiente. O sistema de monitoramento que será utilizado em conjunto com o MultiS deve ser capaz de monitorar os dados do ambiente e abstrair sua distribuição permitindo que o sistema de reconhecimento do contexto acesse-os como se estivessem disponíveis localmente.

Outra importante característica do sistema de monitoramento é a capacidade de notificar o MultiS quando um sensor tiver seu valor modificado, já que é essa

notificação que dispara o evento de recálculo dos grafos de inferência de contexto no MultiS. Entretanto, o MultiS possui um mecanismo de verificação de modificações no ambiente que pode ser utilizado caso o sistema de monitoramento não seja capaz de realizar esta notificação. A capacidade de notificação é, portanto, uma característica importante, mas não obrigatória do sistema de monitoramento usado em conjunto com o MultiS.

4.6.2 Sistema de Adaptação

O sistema de adaptação tem a responsabilidade de efetuar a transformação do comportamento da aplicação em caso de modificação no ambiente. Para funcionar em conjunto com o MultiS, ele deve poder ser notificado dessas modificações. Além disso, a carga dinâmica do código responsável pelo novo comportamento da aplicação e sua instalação é de responsabilidade do sistema de adaptação.

4.7 Aspectos de *Pervasividade*

Durante o Capítulo 3 foram estudados diversos *middlewares* para tratamento do contexto. Características importantes para qualquer sistema que se proponha a resolver problemas relativos à Computação *Pervasiva* foram analisadas no final do capítulo. Essas características, que motivaram os requisitos estabelecidos para este projeto, são resgatadas abaixo, juntamente com a solução correspondente adotada no projeto MultiS.

- **Facilidade de desenvolvimento de aplicações**: Segundo o modelo proposto neste trabalho, existe uma separação entre o papel do programador do filtro e o do programador da aplicação. Esse último não precisa ter conhecimento sobre os detalhes de implementação dos filtros, basta que saiba como utilizá-los. Nesse ponto de vista, os programadores de aplicação podem ser considerados usuários dos filtros, já que esses podem ser utilizados como bibliotecas, liberando os programadores de aplicação de terem conhecimento sobre detalhes de implementação de cada um dos filtros.
- **Compartilhamento de informações**: O MultiS possibilita o compartilhamento de informações tratadas entre as aplicações. Em ambientes onde diversas aplicações utilizam as mesmas informações de contexto, a economia de recursos obtida com esse compartilhamento pode possibilitar que o sistema atenda a mais requisições.
- **Suporte à mobilidade**: Uma das principais características da mobilidade é a possibilidade sempre presente de desconexão. Ela pode afetar não só as aplicações como também os nodos onde se localizam os sensores. Através do mecanismo de tratamento de valores indeterminados, o MultiS oferece uma maneira de tratar os dados indisponíveis. Outros aspectos de mobilidade, como a adaptação do sistema para operação desconectada, devem ser tratados por outras camadas do *middleware* e fogem do escopo do servidor de contexto.
- **Flexibilidade e extensibilidade:** Novos filtros podem ser adicionados e disponibilizados às aplicações em tempo de execução. Essa característica permite que o sistema evolua durante sua utilização e seja capaz de tratar novos sensores e produzir novas informações.

- **Reatividade a modificações no ambiente**: O MultiS permite que aplicações se inscrevam para serem notificadas de modificações no ambiente. Como esse modelo evita a necessidade de aplicações realizarem sucessivas consultas para verificar o estado do ambiente, é ideal para aplicações com características de reatividade ao ambiente.

5 INTEGRAÇÃO AO ISAM E TESTES DE IMPLEMENTAÇÃO

Este capítulo apresenta a implementação do MultiS e sua integração ao EXEHDA, *middleware* de execução do projeto ISAM. Posteriormente apresenta-se o PerMuseum, uma aplicação *Pervasiva* que busca enriquecer a experiência dos usuários durante uma visita a um Museu. Através do projeto e implementação do protótipo dessa aplicação busca-se demonstrar o uso das características propostas por este trabalho. No final do capítulo são apresentados resultados dos testes de execução da aplicação PerMuseum.

5.1 Diagrama de Classes: por Dentro do MultiS

A Figura 5.1 apresenta o diagrama de classes (BOOCH, 1999) da implementação do MultiS já integrado à implementação atual do projeto ISAM. Diversos elementos da arquitetura do MultiS foram omitidos nesse diagrama (*eg.* os módulos `Parser e Verifier`). Buscando manter a objetividade, apenas as classes essenciais ao entendimento do funcionamento do sistema são apresentadas no diagrama, que tem por objetivo apresentar o mecanismo de inferência de contexto utilizado na implementação do MultiS.

A interface `ContextElement` é implementada por duas classes, a classe `Sensor` e a classe `Context`. A primeira é a representação do MultiS para uma informação oriunda diretamente de um sensor do ambiente. A segunda representa um valor de contexto que já foi previamente processado por um filtro e foi, portanto, gerado pelo próprio MultiS. A implementação da interface permite que o sistema de cálculo de informações de contexto acesse esses dois tipos de informação indistintamente. Enquanto o método `getValue` da classe `Sensor` retorna o próprio valor do sensor, a implementação da classe `Context` calcula o valor do objeto utilizando o filtro associado a esse objeto e os objetos `ContextElements` (que podem representar sensores ou outros objetos `Context`). Dessa forma calcula-se recursivamente o valor de um grafo de inferência do contexto.

Os filtros do sistema devem implementar a interface `ContextFilter`, que garante a existência de um método responsável pela filtragem e que é utilizado para produção de valores de contexto. Na Figura 5.1 são apresentados dois exemplos de filtros: `FilterIntegerSum e Filter RoomSensor`.

As classes `JavaRuntimeMonitor e RoomMonitor` são exemplos de `Monitores` [YAMIN, 2004], unidades de *software* que são parte do EXEHDA, *middleware* de execução do projeto ISAM. Esses elementos são responsáveis pelo

monitoramento e gerenciamento dos sensores. A classe `Sensor` é uma abstração do MultiS para um sensor real e é fornecida pelos Monitores.

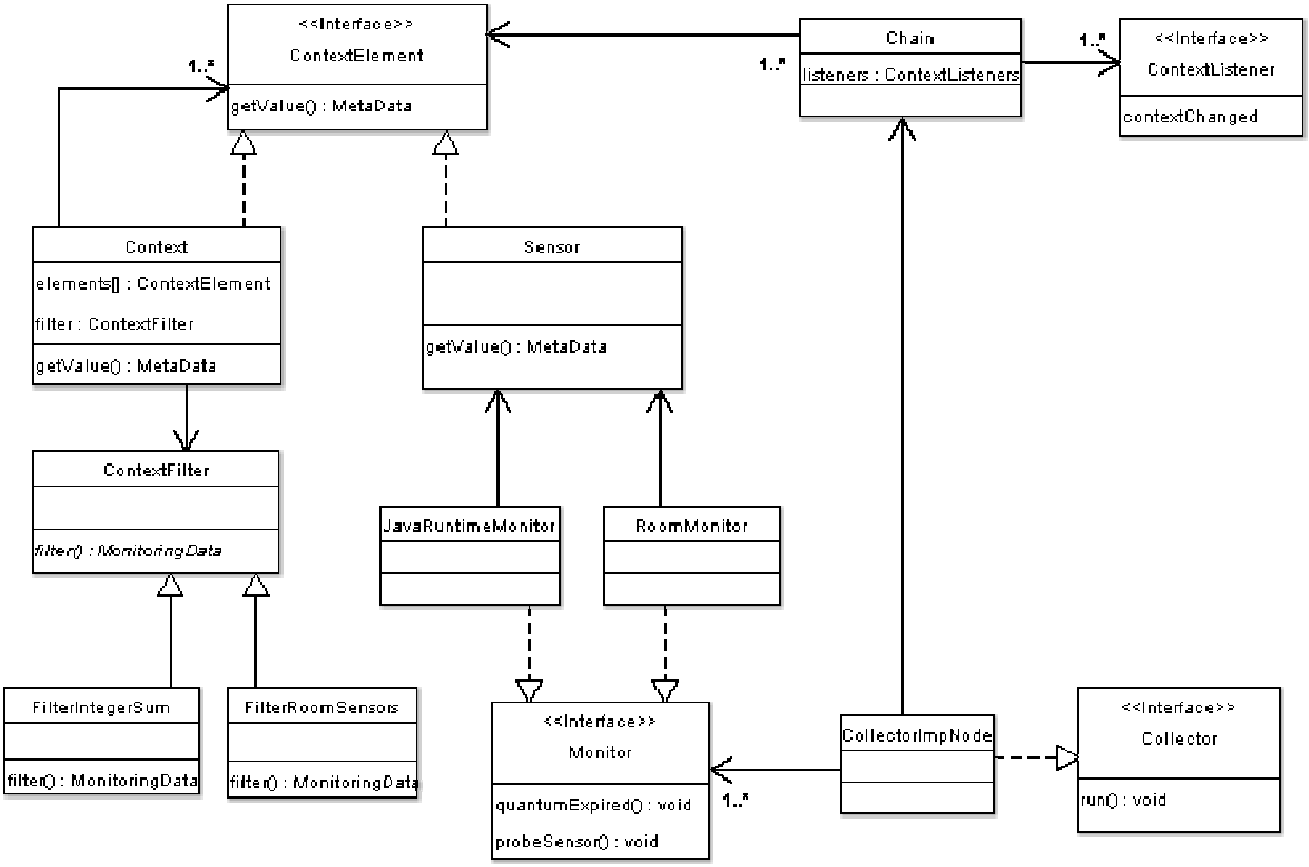

Figura 5.1: Diagrama de classes da implementação do MultiS.

A interface `ContextListner` é implementada por elementos (omitidos no diagrama) que se inscrevem para notificação no MultiS. Permitindo a reatividade do sistema a modificações no ambiente. Um `Chain` é associado a um `ContextElement` e é responsável pela detecção de modificação em seu valor. Além disso, cada objeto `Chain` possui uma lista de `ContextListners` associados. Quando uma modificação no valor de um `ContextElement` é verificada, os `ContextListners` associados ao `Chain` são ativados, iniciando o processo de adaptação.

A classe `CollectorImpNode` faz parte do *middleware* de execução EXEHDA e é responsável por orquestrar todo esse processo, recebendo dos Monitores as notificações de modificação no valor dos sensores e ativando o processo de cálculo das variáveis de contexto.

5.2 Integração do MultiS ao ISAM

O ISAM possibilita que os pré-requisitos à execução do MultiS sejam atendidos, já que possui um Serviço de Monitoramento e um Serviço de Adaptação. O Serviço de Monitoramento do ISAM permite ainda que a instalação de sensores seja feita sob demanda, conferindo agilidade ao Servidor de Contexto. O sistema de adaptação do

ISAM também é adequado, já que possibilita a instalação de código de execução sem necessidade de reinicialização da aplicação.

A integração a um *middleware* de execução completo como o ISAM permite que o MultiS utilize diversos serviços já disponíveis neste *framework*. Os sensores, por exemplo, podem estar distribuídos em diversos computadores, mas o ISAM permite que o MultiS acesse as informações como se elas estivessem disponíveis localmente, abstraindo o acesso e o tempo de acesso a esses dados.

5.3 Estudo de Caso - PerMuseum: Uma Aplicação *Pervasiva*

Um dos usos mais promissores da Computação *Pervasiva* é enriquecer as aplicações tradicionais já existentes. As características da consciência do contexto oferecem novas maneiras de interação com os usuários. A Computação Consciente do Contexto já foi utilizada, por exemplo, para selecionar conteúdo digital relacionado a uma localidade turística (CHEVERST, 2002), (LONDSDALE, 2005). Dentro desta perspectiva, e com o objetivo de validar a implementação do modelo de servidor de contexto proposto, foi desenvolvido o protótipo de um *software* chamado PerMuseum (*Pervasive Museum*).

5.3.1 Aplicação Atual

Diversos museus, como o Guggeinheim de Nova York (GUGGENHEIM, 2006) e o Malba de Buenos Aires (MALBA, 2006), disponibilizam aos seus visitantes o serviço de visita guiada por áudio (*audio guide*). O usuário utiliza um dispositivo semelhante a um PDA com fones de ouvido e um pequeno teclado. Existe, armazenado nesse dispositivo, informações com explicações e análises sobre diversas obras do museu. As obras apresentam, junto ao título, um número que identifica a obra e que deve ser digitado pelo usuário para que o áudio correspondente àquele recurso seja apresentado.

A Figura 5.2 retirada do sítio do museu Metropolitan (METROPOLITAN, 2006) apresenta uma descrição textual onde é possível observar o tipo de material disponibilizado em áudio aos usuários através do dispositivo.

Toda vez que o usuário se locomove e transfere sua atenção de uma obra à outra, é necessário que o código correspondente à sua obra de interesse seja inserido. Esta necessidade de intervenção constante retira o foco do usuário do seu objeto de interesse, desestimulando a utilização dos recursos disponíveis.

A Computação Consciente do Contexto pode ser utilizada para enriquecer a experiência do usuário na utilização desse aplicativo, apresentando, automaticamente e sem a necessidade de nenhuma ação explícita do usuário, o conteúdo disponível sobre a obra observada pelo visitante de acordo com as características do contexto de cada usuário. Diversas informações podem ser utilizadas na seleção do melhor conteúdo a ser disponibilizado pelo visitante.

Pode-se inferir qual obra está sendo observada por um usuário com base em sua localização dentro do museu. Além da localização, diversas outras informações do contexto podem ser utilizadas na seleção. O conteúdo mais adequado ao dispositivo pode ser selecionado com base nos formatos de dados suportados, características de uso, perfil do usuário, etc. Por exemplo, se o dispositivo não tiver suporte à reprodução sonora, ou se nenhum fone estiver conectado no momento, seleciona-se para apresentação apenas informações visuais, como textos e figuras.

Figura 5.2: Exemplo de material disponibilizado em guias eletrônicos de museus (METROPOLITAN, 2006).

Dados a respeito do usuário podem ser fonte importante de informação para a adaptação do sistema. Informações como os idiomas compreendidos pelo visitante, preferências pessoais ou deficiências físicas podem ser utilizadas pelo sistema para uma seleção personalizada de conteúdo. Para deficientes visuais, por exemplo, não seriam disponibilizados conteúdos em que esse sentido fosse necessário.

Muitos museus, embora não tenham atualmente o serviço de guia de áudio, possuem material digital sobre suas obras. Por exemplo, o MARGS (MARGS, 2006) disponibiliza um CD-ROM com informações a respeito de diversas obras de arte. Essas informações poderiam ser utilizadas no PerMuseum. Para isso, material a ser disponibilizado precisaria ser classificado conforme suas características de acordo com os diferentes segmentos de visitantes que utilizarão o sistema. Sugere-se a catalogação dessas informações como trabalho futuro.

5.3.2 Comportamento da Aplicação PerMuseum

A primeira etapa na utilização do sistema é o rápido preenchimento de um formulário, onde o usuário informa os idiomas que podem ser utilizados pelo sistema para apresentação de conteúdo. Esta informação será utilizada para geração de dados do sensor de idiomas.

A aplicação então realiza uma requisição, utilizando CD-XML, ao sistema de reconhecimento de contexto, inscrevendo-se para ser notificada em caso de modificação

no ambiente. A partir desse momento, o sistema está em funcionamento e pronto para adaptar-se quando necessário.

A locomoção do visitante dentro do museu altera o valor retornado pelo sensor de localização. Verifica-se, então, se a atenção do visitante foi transferida para outro objeto do museu. Caso esta modificação seja confirmada, busca-se por conteúdo relativo ao novo objeto de interesse, que é selecionado de tal forma que seja adequado às características relevantes ao contexto do usuário.

Paralelamente à monitoração da localização do usuário, o sistema procura por eventos do museu que possam ser de interesse do visitante. Esses eventos podem ser relativos a uma visita guiada ou a uma atração do museu (*eg*. cinema, planetário, palestra). Caso um evento classificado como interessante ao usuário seja detectado, notifica-se o usuário, que então recebe detalhes sobre a localização e horário do evento.

5.3.3 Requisitos para Implementação

O MultiS funciona em conjunto com diversos outros sistemas, conforme detalhado no Capítulo 4. Para a implementação do protótipo do *software* do PerMuseum, alguns desses sistemas foram emulados.

A monitoração dos dados do ambiente, por exemplo, é uma tarefa externa ao MultiS e está fora do escopo deste trabalho. Ainda assim, as informações dos sensores são necessárias aos testes de execução realizados. Portanto, durante os testes, foi utilizada a camada de monitoramento artificial, onde os dados dos sensores foram inseridos manualmente.

Os sensores necessários à execução da aplicação são listados na Tabela 5.1, onde também se encontra sua descrição. Algumas informações, como localização, são obtidas a partir de sensores no ambiente, outras, como os idiomas do usuário, podem ser obtidas através de um rápido preenchimento de formulário pelo visitante.

Algumas informações de contexto são específicas de cada usuário, enquanto outras são comuns a todos os usuários e podem ser compartilhadas entre suas aplicações. No PerMuseum os dados que são específicos do usuário podem ser identificados por um sufixo na formação no nome da informação. As informações são identificadas da seguinte forma: `<nome genérico do sensor>_<identificador do usuário>`. Por exemplo, o sensor `LocalizacaoAtualUsuario`_0331 fornece informações sobre a localização do usuário identificado pela chave numérica `0331`. Embora não exista nenhuma restrição quanto aos nomes utilizados na formação dos sensores e demais elementos de contexto, aconselha-se a utilização de nomes segundo esse padrão, que objetiva facilitar a identificação dos sensores específicos pelo programador de aplicação.

Os dados dos sensores listados anteriormente são processados pelos filtros. Os filtros especificados na Tabela 5.2 são necessários à execução da aplicação PerMuseum e devem estar disponíveis no repositório para a correta execução da aplicação. Esses filtros são parte do conjunto básico de bibliotecas de filtros disponibilizados com o MultiS.

Tabela 5.1: Sensores necessários à execução do PerMuseum.

Nome do Sensor	Informação disponibilizada
`LocalizacaoAtualUsuario_0331`	Sensor que retorna informações sobre a localização corrente do dispositivo no formato de coordenadas, latitude, longitude e andar atual. Existe um sensor do mesmo tipo para cada dispositivo em execução.
`RecursosMuseu`	Disponibiliza um mapa do museu contendo a localização de todos os recursos.
`MaterialDisponivel`	Lista todas as informações disponíveis para as obras em exibição no museu. Cada elemento da lista contém o tipo de dispositivo necessário à sua apresentação e o idioma da informação.
`TipoDispositivoUsuario_0331`	Retorna as características do dispositivo do usuário, assim como os formatos de dados suportados por esse dispositivo. Existe um sensor do mesmo tipo para cada dispositivo em execução.
`LocalizacaoGuia`	Sensor que retorna informação sobre a localização corrente do dispositivo do guia no formato de coordenadas, latitude, longitude e andar atual.
`IdiomasPalestrante`	Lista de idiomas dominados pelo palestrante.
`IdiomasUsuario_0331`	Lista de idiomas compreendidos pelo usuário. Existe um sensor do mesmo tipo para cada usuário.
`Atracao`	Disponibiliza informação sobre eventos do museu, contém um horário associado, o idioma em que o evento será suportado e uma identificação sobre o tipo de mídia disponibilizada no evento.

Tabela 5.2: Filtros necessários à execução do PerMuseum.

Nome do Filtro	Tratamento Realizado
`FiltroEscolheElementoPrioridade`	Retorna apenas um registro. O registro com valor mais novo tem prioridade mais alta. Se dois registros tiverem sido atualizados em tempo semelhante, o desempate se dá pelo parâmetro do filtro.
`FiltraListaPorCriterios`	Filtra uma lista retornando apenas os elementos cujas propriedades estiverem contidas nos outros dados do filtro. As propriedades dos parâmetros que são verificadas devem ser informadas nos parâmetros do filtro.
`FiltroIdentificaElementoEmMapa`	Este filtro recebe duas informações: um mapa contendo recursos e suas localizações, e uma localização nesse mapa. O filtro retorna o recurso do mapa mais próximo à posição informada.
`FiltroAgregador`	Retorna uma lista com base na mescla de outras listas.
`FiltroAgregadorDadoEmLista`	Agrega uma informação a todos os elementos de uma lista. O parâmetro especifica a propriedade dos elementos da lista que será modificada.

5.3.4 Dados de Contexto Produzidos

Dados de contexto são informações geradas pelo MultiS com base nos dados dos sensores e no processamento dos filtros. Enquanto algumas dessas informações podem ser compartilhadas entre diversas instâncias da aplicação que estão utilizando o sistema, outras são relativas a um determinado usuário, e, portanto não são compartilhadas entre aplicações de diferentes usuários. Os dados de contexto produzidos durante a execução do PerMuseum são listados na Tabela 5.3.

Tabela 5.3: Dados de contexto produzidos pelo MultiS durante a execução do PerMuseum.

`MaterialObraParaUsuario_0331`	Informação composta por uma lista de materiais sobre a obra de interesse de um determinado usuário. A lista contém apenas materiais nos idiomas aceitos e cujo material é suportado pelo dispositivo do usuário.
`MaterialSobreObra_0331`	Lista de Materiais disponíveis sobre a obra de interesse do usuário.
`ObraDeInteresse_0331`	Identificador da obra de interesse do usuário.
`MateriaisEvento`	Lista de materiais sobre eventos (palestras ou atrações) programados no museu.
`EventosParaUsuario_0331`	Lista de materiais sobre eventos (palestras ou atrações) programados no museu. A lista contém apenas Materiais nos idiomas aceitos e cujo material é suportado pelo dispositivo do usuário.
`PalestraSobreObra`	Identificador de obra sobre a qual uma palestra está em andamento.
`Palestra`	Identificador de obra sobre a qual uma palestra está sendo realizada associada aos idiomas utilizados na palestra.
`MateriaisAtracao`	Lista de todos os materiais disponíveis para uma determinada atração.
`MaterialApresentado_0331`	Indica qual contexto final da aplicação do usuário e, portanto qual informação deve ser exibida por sua aplicação.

5.3.5 Composição dos Sensores com utilização de CD-XML

A Figura 5.3 mostra o código CD-XML utilizado pela aplicação PerMuseum para inscrição junto ao sistema de reconhecimento de contexto MultiS. Na Figura 5.4 essa requisição é representada graficamente. É possível perceber os filtros que produzem cada uma das informações de contexto e quais são os sensores utilizados por cada filtro.

```
<?xml version="1.0" encoding="UTF-8"?>
<context label="MaterialApresentado_0331"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="cd-xml.xsd">
    <filter name="FiltroEscolheElementoPrioridade">
        <parameters>
            <parameter>evento</parameter>
        </parameters>
    </filter>
    <context label="MaterialObraParaUsuario_0331">
        <filter name="FiltraListaPorElemento">
            <parameters>
                <parameter>tipoDispositivo</parameter>
                <parameter>idiomas</parameter>
            </parameters>
        </filter>
        <context label="MaterialSobreObra_0331">
            <filter name="FiltraListaPorElemento">
                <parameters>
                    <parameter>obra</parameter>
                </parameters>
            </filter>
            <sensor>MaterialDisponivel</sensor>
            <context label="ObraDeInteresse_0331">
                <filter name="FiltroIdentificaElementoEmMapa">
                    <parameters></parameters>
                </filter>
                <sensor>LocalizacaoAtualUsuario_0331</sensor>
                <sensor>RecursosMuseu</sensor>
            </context>
        </context>
        <sensor>TipoDispositivoUsuario_0331</sensor>
        <sensor>IdiomasUsuario_0331</sensor>
    </context>
    <context label="EventosParaUsuario_0331">
        <filter name="FiltraListaPorElemento" default="">
            <parameters>
                <parameter>tipoDispositivo</parameter>
                <parameter>idiomas</parameter>
            </parameters>
        </filter>
        <context label="MateriaisEvento">
            <filter name="FiltroAgregador">
                <parameters></parameters>
            </filter>
            <context label="Palestra">
                <filter name="FiltroAgregadorDadoEmLista">
                    <parameters>
                        <parameter>idioma</parameter>
                    </parameters>
                </filter>
                <context label="PalestraSobreObra">
                    <filter name="FiltroIdentificaElementoEmMapa">
                        <parameters></parameters>
                    </filter>
                    <sensor>LocalizacaoGuia</sensor>
                    <sensor>RecursosMuseu</sensor>
                </context>
```

```
                <sensor>IdiomasPalestrante</sensor>
            </context>
            <context label="MateriaisAtracao">
                <filter name="FiltraListaPorElemento">
                    <parameters>
                        <parameter>atracao</parameter>
                    </parameters>
                </filter>
                <sensor>Atracao</sensor>
                <sensor>MaterialDisponivel</sensor>
            </context>
        </context>
        <sensor>TipoDispositivoUsuario_0331</sensor>
        <sensor>IdiomasUsuario_0331</sensor>
    </context>
</context>
```

Figura 5.3: CD-XML utilizado na aplicação PerMuseum.

Na Figura 5.4, alguns nodos são representados por um círculo escurecido. Esses elementos representam informações genéricas (*ie.* não são específicas de um determinado usuário). Essas informações podem ser compartilhadas entre todas as aplicações em execução no ambiente do museu.

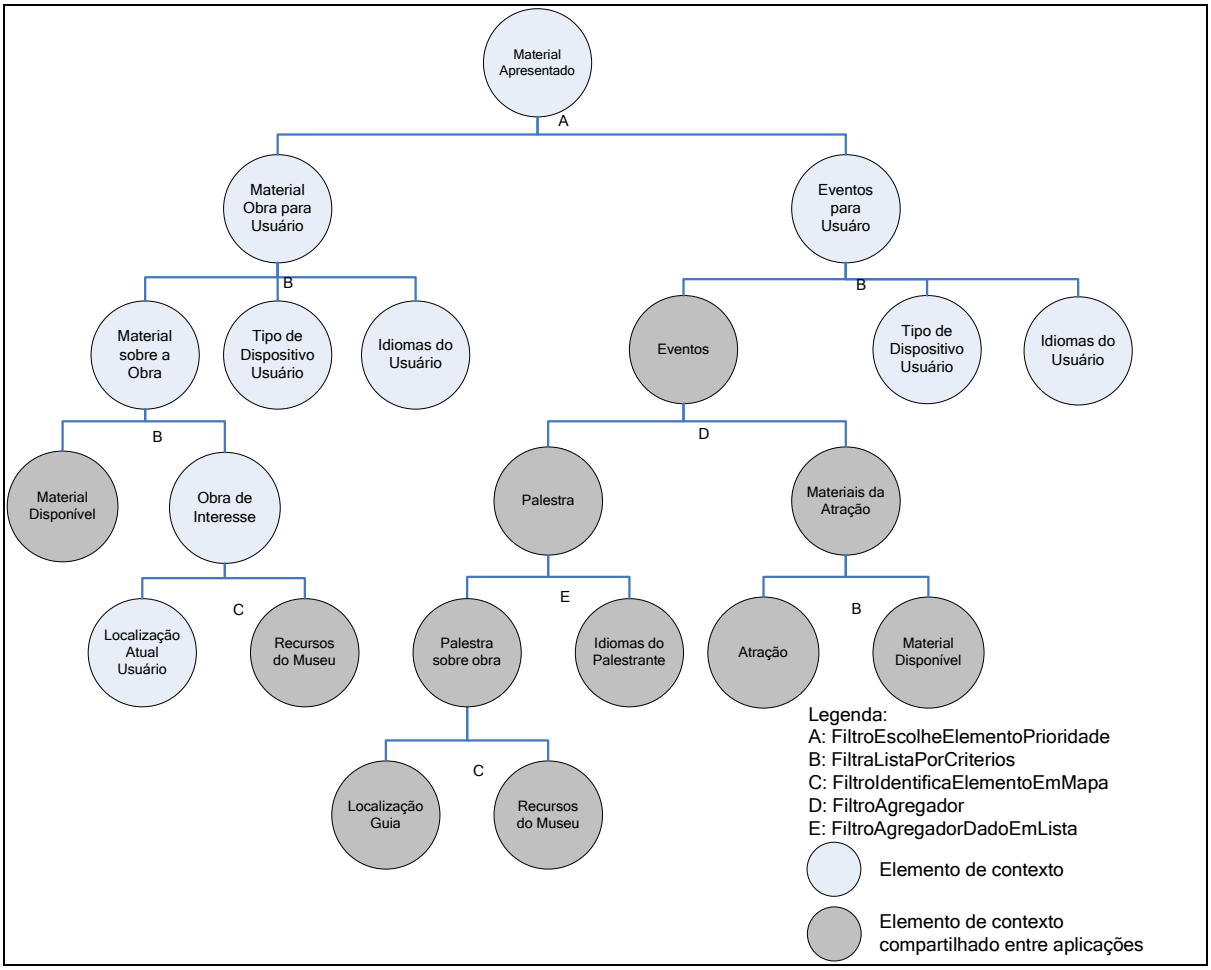

Figura 5.4: Representação gráfica do trecho CD-XML utilizado no PerMuseum.

5.4 Testes

Nesta seção apresenta-se dois conjuntos de testes da aplicação PerMusem sobre a implementação existente do MultiS. O primeiro conjunto de testes verifica como a aplicação reage às modificações que ocorrem no ambiente. O segundo conjunto de testes compara o comportamento do sistema com e sem o compartilhamento de informações de contexto entre aplicações.

5.4.1 Teste de Reatividade a Modificações no Ambiente

O gráfico da Figura 5.5 demonstra o comportamento do protótipo da aplicação PerMuseum em um teste de execução. Os sensores foram alimentados com dados de testes. No teste correspondente a essa figura, apenas o sensor `LocalizacaoAtualUsuario_0331` teve seu valor modificado ao longo do tempo. Cada modificação na localização do usuário ocasionou um recálculo em toda uma sub-árvore de inferência de contexto, e, eventualmente, resultando na modificação do tipo de material apresentado ao usuário. Na Figura 5.5, o eixo y representa o tipo de material apresentado ao usuário, identificado por um número. As linhas horizontais representam o comportamento da aplicação em cada instante da execução. As linhas verticais representam o momento em que o sensor tem seu valor modificado.

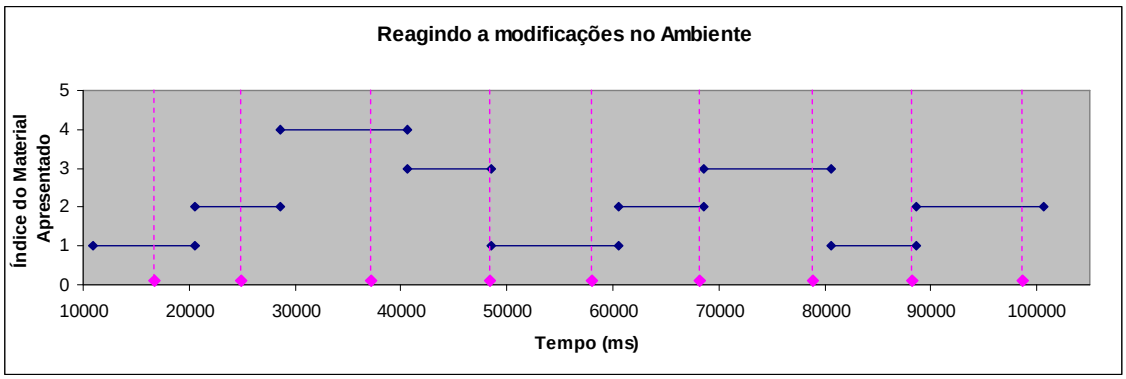

Figura 5.5: Tempos entre modificação no ambiente e reação do sistema.

Pode-se identificar que a aplicação inicia a execução apresentando o material identificado pelo índice 1, em seguida, ocorre uma modificação no valor do sensor de localização, que em alguns momentos resulta em transformação no comportamento da aplicação, que passa a apresentar o material identificado pelo índice 2.

O sistema de monitoramento utilizado nesse teste foi emulado, e não apresenta a capacidade de notificação de modificações no valor dos sensores. Foi utilizado, portanto o mecanismo identificação de modificação do próprio MultiS, que é feito por consultas periódicas. Nesse caso, as consultas eram feitas a cada segundo. Quando a modificação no sensor aconteceu próximo ao final desse tempo, como na quarta modificação do sensor, o sistema reagiu rapidamente. Já quando a modificação no valor do sensor aconteceu próxima ao início do tempo, o sistema demorou mais tempo para reagir à modificação. O tempo de reação do sistema inclui o tempo necessário para calcular o valor das árvores de inferência do contexto que precisam ser recalculadas depois da alteração de um sensor.

Inúmeros testes foram realizados sob essas circunstâncias. O sistema apresentou sempre um comportamento uniforme. O gráfico da Figura 5.5 representa apenas uma dessas execuções.

5.4.2 Teste de Desempenho do Compartilhamento de Informações em Árvore

O objetivo deste teste é demonstrar como o compartilhamento de sub-árvores pode impactar no desempenho do sistema em uma situação real de uso. Para tanto, dois tipos de testes foram realizados: o primeiro sem o compartilhamento de sub-árvores e o segundo com esse compartilhamento. O critério de desempenho utilizado é o tempo necessário para que todas as árvores de inferência existentes no servidor de contexto sejam processadas após uma modificação no ambiente.

Cada operação de filtragem consome certo tempo de processamento. O tempo total para o processamento de todas as árvores de inferência existentes, portanto, tende a crescer proporcionalmente ao número de aplicações inscritas no servidor de contexto. Esse tempo total tem impacto direto sobre a quantidade de aplicações suportadas por uma instância do servidor de contexto. Se o tempo de processamento for grande demais, a aplicação irá demorar a adaptar-se ao novo estado do ambiente.

A Tabela 5.4 apresenta o resultado dos testes de execução da aplicação PerMusem sem compartilhamento de informações entre as árvores de inferência de diferentes aplicações. As aplicações foram executadas com um intervalo de atualização dos sensores de 1 segundo. Os resultados foram calculados com base em 20 execuções para cada quantidade de aplicações inscritas no servidor de contexto em dado instante. Os tempos das execuções individuais podem ser encontrados no APÊNDICE B.

Tabela 5.4: Dados sobre execução dos testes sem compartilhamento de sub-árvores.

Número de Aplicações	1	5	10	20	50	100
Tempo médio	12.7	49.35	79.8	179.45	252.75	957.8
Tempo médio por aplicação	12.7	9.87	7.98	8.9725	5.055	9.578
Menor Tempo	2	6	20	62	112	225
Maior Tempo	26	137	176	613	762	2528
Desvio padrão	6.14817046	34.71732	38.71706	145.3698	152.3918	841.0698

Tempos em ms.

Pode-se perceber pelos resultados dos testes que o servidor de contexto demonstrou uma degradação no desempenho nos casos de 100 aplicações executando concorrentemente. Houve casos onde o processamento de todas as árvores das aplicações inscritas demorou mais tempo que o intervalo de atualização dos sensores. Ou seja, novas informações sobre os sensores já estavam disponíveis no servidor de contexto enquanto os dados da atualização anterior ainda estavam sendo processados. Isso causou uma degradação ainda maior no desempenho do sistema, que resultou por gerar tempos muito altos para algumas execuções (*ie.* uma diferença muito grande entre os tempos mínimos e máximos). A degradação também pode ser percebida pelo aumento do desvio padrão dos tempos das execuções. Nesse caso, pode-se dizer que o sistema estava próximo ao seu limite de utilização.

A Tabela 5.5 contém os resultados obtidos com testes de execução da aplicação PerMuseum com o compartilhamento de sub-árvores. Com o aproveitamento de alguns dados já processados, é possível perceber que os tempos de execução para o processamento de todas as árvores de inferências das aplicações inscritas no servidor de contexto foram bem menores. Além disso, não ocorreu a degradação de performance quando os testes foram executados com 100 aplicações inscritas.

Tabela 5.5: Dados sobre a execução dos testes com compartilhamento de sub-árvores.

Número de Aplicações	1	5	10	20	50	100
Tempo médio	11.15	37.85	44.55	106.7	137.3	254.85
Tempo médio por aplicação	11.15	7.57	4.455	5.335	2.746	2.5485
Menor Tempo	1	2	12	29	51	134
Maior Tempo	33	98	150	221	251	529
Desvio padrão	9.404786016	29.61201	35.8263	54.07899	53.78329	125.6022

Tempos em ms.

O desvio padrão obtido com os tempos de execução também manteve-se dentro dos limites da normalidade. Em nenhum dos testes realizados o tempo de processamento ultrapassou o tempo de atualização dos sensores.

Pode-se concluir, comparando as informações nas duas tabelas, que com o compartilhamento de sub-árvores de inferência de contexto o MultiS teria a possibilidade de suportar a inscrição de mais aplicações do que sem esse compartilhamento.

Essas informações foram colhidas em um ambiente controlado, onde apenas um tipo de aplicação era executada e onde todas as aplicações continham informações passíveis de serem compartilhadas. Destaca-se que o ganho de performance obtido com o compartilhamento de sub-árvores é diretamente relacionado ao tipo de aplicação em execução em cada ambiente.

5.5 Comparação entre o MultiS e Outros Projetos

No final do Capítulo 3 foi apresentada uma tabela onde foram comparados os principais *middlewares* existentes para tratamento de contexto. Esta seção tem por objetivo inserir o MultiS nessa comparação. Para tanto, discute-se para cada tópico da comparação, a solução proposta por este trabalho. Por fim, na Tabela 5.6, repete-se a Tabela 3.1 com a inclusão de uma coluna referente ao MultiS.

O MultiS é capaz de produzir dados de contexto com base em informação de múltiplos sensores utilizando as árvores de inferência de contexto. Os dados produzidos podem ser utilizado por diversas aplicações através do compartilhamento de sub-árvores de contexto. O código da aplicação é independente de qualquer tratamento realizado sobre os dados do contexto. Esse acoplamento é realizado através da utilização da linguagem CD-XML. Essa linguagem disponibiliza um mecanismo para tratamento de valores indeterminados que pode ser utilizado para tratar casos de desconexão, que é uma característica da mobilidade.

As aplicações podem obter dos dados de contexto através de consultas explícitas ou através do mecanismo de inscrição/notificação, que é mais utilizado para programar a reatividade ao ambiente em aplicações conscientes do contexto. O tratamento das informações é realizado através de componentes denominados filtros, que podem ser parametrizados para atender a demandas mais específicas de cada aplicação. Esses elementos podem ser recombinados através da linguagem CD-XML, liberando o programador de aplicação dos detalhes de implementação desse tratamento. A Tabela 5.6 abaixo insere o MultiS na comparação entre os *middlewares* existentes para reconhecimento de contexto.

Tabela 5.6: Comparação entre o MultiS e os *middlewares* existentes para reconhecimento de contexto.

	Context Toolkit	Solar	Aura/ CIS	JCAF	EgoSpaces	MultiS
Compartilhamento dos dados monitorados	X	X	X	X	X	X
Capaz de integrar informações de diferentes sensores num mesmo dado tratado	X	X	X	X	-	X
Compartilhamento dos dados tratados	X	X	-	X	X	X
Independência entre tratamento e aplicação	-	X	X	-	-	X
Suporte à mobilidade / desconexão	-	-	X	-	X	X
Capacidade de monitoramento /tratamento sob demanda	X	X	X	X	X	X
Capacidade de monitoramento /tratamento do tipo inscrição/notificação	X	X	-	X	-	X
Tratamento parametrizável, adaptável a cada aplicação	-	-	X	X	-	X
Capaz de compor novos dados de contexto sem necessidade de programação	-	X	X	-	-	X

6 CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo resgata os tópicos de pesquisa considerados no desenvolvimento deste trabalho. Os requisitos de um sistema de reconhecimento de contexto voltado à Computação *Pervasiva* são resgatados, assim como a proposta apresentada neste pelo MultiS para cada um desses problemas. Posteriormente discutem-se oportunidades de trabalhos futuros identificados durante o desenvolvimento do MultiS.

6.1 Conclusões

Observou-se nos últimos anos um crescente desenvolvimento e popularização de dispositivos móveis. PDAs e celulares com suporte a protocolos como *Bluetooth* ou WiFi, que possibilitam o acesso a elementos do ambiente, estão cada vez mais comuns. Segundo (ANDREW, 2006), não importa o tipo de dispositivo móvel disponível hoje, a revolução do *wireless* está apenas começando. Acredita-se que os próximos passos evolutivos serão em direção à integração entre esses dispositivos móveis e o ambiente computacional tradicional.

Esse é o cenário onde a Computação Consciente do Contexto ganha importância, capacitando os dispositivos móveis a reconhecer e adaptarem-se aos recursos disponíveis no ambiente em cada instante. Um dos grandes desafios impostos por essa proposta, é a instrumentação dos programadores.

Hoje, cada programador que deseja desenvolver uma aplicação consciente do contexto deve programar todas as etapas do processo adaptativo: monitoramento, reconhecimento do contexto e adaptação. Isso insere um custo muito grande ao desenvolvimento e desestimula a criação desse tipo de aplicação. A utilização de *frameworks* e *middleware*, que provêem os serviços de infra-estrutura utilizados no processo de consciência do contexto, pode facilitar esse processo e permitir que o programador preocupe-se apenas com o desenvolvimento da aplicação (DEY, 2000).

Encontram-se em desenvolvimento diversos trabalhos que podem ser utilizados como suporte à execução desse tipo de aplicação. No que se refere ao tratamento do contexto, entretanto, nenhum dos trabalhos estudados atende às principais características necessárias a um sistema de reconhecimento de contexto voltado às aplicações *pervasivas*.

O MultiS, preenche esse espaço existente, propondo um sistema de reconhecimento de contexto voltado às características das aplicações *pervasivas* conscientes do contexto. A integração de informações de diferentes sensores é possível com a utilização de CD-XML, linguagem simples que pode, inclusive, ser programada através de ferramentas de desenvolvimento populares. Essa linguagem oferece recursos para

tratamento de exceções, que pode ser usado para tratamento de desconexão, problema freqüente em ambientes com elementos móveis.

Através do mecanismo de inscrição/notificação, o MultiS possibilita que as aplicações reajam a modificações no ambiente. A execução de diversas aplicações concorrentes é suportada pelo servidor de contexto, que busca economia de recursos ao compartilhar informações entre aplicações sempre que possível. Uma síntese dos requisitos e das soluções propostas pelo MultiS pode ser observada na Tabela 6.1.

Tabela 6.1: Requisitos de um sistema de reconhecimento de contexto orientado à Computação *Pervasiva*.

Requisito	**Solução adotada no projeto MultiS**
Facilidade de desenvolvimento de aplicações	Reutilização de filtros e tratadores de informação entre as aplicações
Suporte à mobilidade	Capaz de lidar com desconexão e mobilidade através do tratamento de valores indeterminados
Flexível e extensível	Novos elementos podem ser adicionados em tempo de execução
Reativo a modificações no ambiente	Consultas do tipo inscrição/notificação

6.2 Trabalhos Futuros

Algumas oportunidades interessantes de trabalhos se apresentaram durante o desenvolvimento desta dissertação. Por limitações de tempo e buscando preservar a objetividade do trabalho, nem todas elas puderam ser desenvolvidas e incluídas no trabalho realizado. Espera-se que esses tópicos motivem novas pesquisas e sejam desenvolvidos e aprofundados em outros trabalhos.

- Extensão da linguagem CD-XML oferecendo mecanismos de procura de sensores por diversos critérios;
- Desenvolvimento de uma quantidade significativa de filtros, cobrindo a maioria das operações realizadas sobre dados de contexto;
- Estudar a questão da privacidade das informações de contexto;
- Estudar o compartilhamento de informações de contexto com definição diferente;

6.2.1 Trabalhos futuros relativos à aplicação PerMuseum

A aplicação PerMuseum apresentada no Capítulo 5 pode ser estendida e aprofundada. Abaixo se encontram algumas oportunidades de trabalhos futuros relativos a essa aplicação.

- Aprofundamento dos testes realizados na aplicação PerMuseum;
- Catalogação do material existente no MARGS e disponibilização dessas informações através do PerMuseum;
- Teste de execução da aplicação PerMuseum em situação real de utilização em um Museu;
- Medição do desempenho da aplicação em uma situação real de utilização;
- Comparação, através de pesquisa de opinião junto aos visitantes, do PerMuseum com a aplicação tradicional existente atualmente;

REFERÊNCIAS

ANDREW, A. The New Age of Wireless. **Scientific American**, [S.l.], v.295, n.4, p.20, Oct. 2006.

ALTMANN, J. et al. Context-Awareness on Mobile Devices - The Hydrogen Approach. HICSS. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 36., 2003, Big Island, Hawaii. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2003.

AUGUSTIN, I.; YAMIN, A.; BARBOSA, J.; SILVA, E.; CAVALHEIRO, G.; GEYER, C. ISAM: um Middleware para Aplicações Móveis Distribuídas. **Revista de Informática Teórica e Aplicada (RITA)**, Porto Alegre, v.8, n. 2, p.41-58, 2001.

AUGUSTIN, I. **Abstrações para uma linguagem de Programação visando Aplicações Móveis Conscientes do Contexto em um Ambiente de Pervasive Computing**. 2003. 194 p. Tese (Doutorado em Ciência da Computação), Instituto de Informática, UFRGRS, Porto Alegre.

FERREIRA, A. B. **Dicionário Aurélio Eletrônico Século XXI.** [S. l.]: Campus,1999.

AURA: Project Aura - Distraction-free Ubiquitous Computing. Disponível em: <http://www-2.cs.cmu.edu/~aura/internal.html>. Acesso em: jan. 2005.

BARBOSA, J. L. V. **Holoparadigma:** um Modelo Multiparadigma Orientado ao Desenvolvimento de Software Distribuído. 2002. 213 f. Tese (Doutorado em Ciência da Computação)- Instituto de Informática, UFRGRS, Porto Alegre.

BARDAM J. E. **The Java Context Awareness Framework (JCAF) -- A Service Infrastructure and Programming Framework for Context-Aware Applications.** Aarhus, Denmark: Centre for Pervasive Computing, 2003. (Technical Report 2004-PB-61).

BCS: The British Computer Society**. Grand Challenges in Computing**. Disponível em <http://www.bcs.org/BCS/Awards/Events/GrandChallenges/conferencereports.htm>. Acesso em: jan. 2005.

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML Guia do usuário.** Rio de Janeiro: Campus, 1999. p. 217-219.

CHEN, G. **Solar**: Building A Context Fusion Network for Pervasive Computing. 2004. Tese (Doutorado) – Dartmouth College, Hanover, New Hampshire, USA.

CHEN, G.; LI M.; KOTZ, D. Design and implementation of a large-scale context fusion network. In: ANNUAL INTERNATIONAL CONFERENCE ON MOBILE AND UBIQUITOUS SYSTEMS: NETWORKING AND SERVICES, MOBIQUITOUS, 1., 2004, Boston, Massachusetts. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2004.

CHEN, G.; KOTZ, D. **Solar**: A pervasive-computing infrastructure for context-aware mobile applications. 2002. (Technical Report TR2002-421). Disponível em: <ftp://ftp.cs.dartmouth.edu/TR/TR2002-421.pdf>. Acesso em: jun. 2004.

CHEN, G.; KOTZ, D. **A Survey of Context-aware Mobile Computing Research.** [S.l.]: Dartmouth Computer Science. Disponível em: <http://www.cs.dartmouth.edu/~solar/>. Acesso em: jul. 2005.

CHEVERST, K.; MITCHEL, K.; DAVIS, N. The role of adaptive hypermedia in a contextaware tourist GUIDE. **Communications of the ACM**, New York, v. 45, n.5, p. 47-51, 2002.

DEY, A. **Providing Architectural Support for Building Context-aware Applications**. 2000.183 f. Tese (Doutorado)- Georgia Institute of Technology, Atlanta, USA.

DEY, A.; ABOWD, G. The Context Toolkit: Aiding the Development of Context-Aware Applications. In: WORKSHOP ON SOFTWARE ENGINEERING FOR WEARABLE AND PERVASIVE COMPUTING, SEWPC, 2000, Limerick, Irlanda. **Proceedings…** [S.l.:s.n.], 2000.

EBLING, M.; HUNT, G.; LEI, H. Issues for Context Services for Pervasive Computing. In: ADVANCED WORKSHOP ON MIDDLEWARE FOR MOBILE COMPUTING, Middleware, 2001, Heidelberg, Germany. **Proceedings…** [S.l.:s.n.], 2001.

THE ENDEAVOUR Expedition**:** Charting the Fluid Information Utility. Disponível em: <http://endeavour.cs.berkeley.edu/>. Acesso em: jan. 2005.

FEHLBERG, F. **Middlewares para Reconhecimento do Contexto na Computação Pervasiva.** 2005. 84 f. Trabalho Individual (Mestrado em Ciência da Computação)- Instituto de Informática, UFRGS, Porto Alegre.

FEHLBERG, F. Definindo e Usando Contexto Derivado de Multi-Sensores. In: ESCOLA REGIONAL DE REDES DE COMPUTADORES, ERRC, 3., 2005, Santa Cruz do Sul. **Anais…** Santa Cruz do Sul: UNISC, 2005. p. 40-46.

FRIDAY, A. et al. Adaptive Applications: The MOST Experience. **Journal of Integrated Computer-Aided Engineering**, [S.l.], p.143- 158, 1999.

FURUKAWA, J.; SIEWIOREK, D.; SMAILAGIC, A. SenSay: A Context-Aware Mobile Phone. In: INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, ISWC, 7., 2003, White Plains, NY, USA. **Proceedings...** Los Alamitos: IEEE Computer Society, 2003. p.248-249.

GAIA, Active Spaces for ubiquitous computing. Disponível em: <http://gaia.cs.uiuc.edu/>. Acesso em: jan. 2005.

GARLAN, D.; SOUSA, J. Aura: An architectural framework for user mobility in ubiquitous computing environments. In: WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE, WICSA, 2002, Montreal, Quebec. **Proceedings…** [S.l.]: Kluwer. 2002.

GARLAN, D.; STEENKISTE, P.; SCHMERL, B. Project Aura: Toward Distractionfree Pervasive Computing**. IEEE Pervasive Computing**, New York, v.1, n.3, 2003.

HENGARTNER, U.; STEENKISTE, P. Access Control to Information in Pervasive Computing Environments. In: WORKSHOP ON HOT TOPICS IN OPERATING SYSTEMS, HotOS, 9., 2003, Hawaii, USA. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2003.

HENRICKSEN, K.; INDULSKA, J.; RAKOTONIRAINY, A. Modeling Context Information in Pervasive Computing Systems. In: INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING, PerCom, 1., 2002. **Proceedings…** London, UK: Springer-Verlag, 2002.

HOFER, T. et al. Context-Awareness on Mobile Devices - the Hydrogen Approach. In: ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES, HICSS, 36., 2003, Big Island, Hawaii. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2003.

JUDD, G.; STEENKISTE, P. Providing Contextual Information to Pervasive Computing Applications. In: IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS, PERCOM, 1., 2003, Fort Worth, Texas. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2003.

JULIEN, C.; ROMAN, G. Egocentric context-aware programming in ad hoc mobile environments. In: SYMPOSIUM ON FOUNDATIONS OF SOFTWARE ENGINEERING, SIGSOFT, 10., 2002, Charleston, South Carolina. **Proceedings…** [S.l.: s.n.], 2002. p. 21-30.

KAPPEL, G. et al. Towards a Generic Customisation Model for Ubiquitous Web Applications. In: INTERNATIONAL WORKSHOP ON WEB ORIENTED SOFTWARE TECHNOLOGY, IWWOST, 2., 2002, Málaga, Spain. **Proceedings…** [S.l.: s.n.], 2002.

LAERHOVEN, K. V.; SCHMIDT, A.; GELLERSEN, H. Multi-Sensor Context Aware Clothing. In: INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, ISWC, 6., 2002, Seattle, Washington. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2002.

LONDSDALE, P.; BEALE, R.; BYRNE, W. Using context awareness to enhance visitor engagement in a gallery space. In: HUMAN-COMPUTER INTERACTION, HCI, 2005, Edinburgh, England. **Proceedings…**[S.l.: s.n.], 2005.

MARGS: Museu de Arte do Rio Grande do Sul Ado Malagoli. Disponível em: <http://www.margs.org.br/_conteudo/home.php>. Acesso em: jun. 2006.

MALBA: Museo de Arte Latinoamericana de Buenos Aires. Disponível em: <http://www.malba.org.ar/web/>. Acesso em: jun. 2006.

METROPOLITAN: The Metropolitan Museum of Art. Disponível em: <http://www.metmuseum.org/>. Acesso em: fev. 2006.

MORAES, M.; SILVA, L.; SCHAEFFER FILHO, A.; GEYER, C.; YAMIN, A; AUGUSTIN, I. A Scalable dissemination service for the ISAM architecture. In: SYMPOSIUM ON COMPUTER ARCHITECTURE AND HIGH PERFORMANCE COMPUTING, SBAC – PAD, 17., 2005, Rio de Janeiro. **Proceedings…** Washington: IEEE, 2005. p.77-84.

OXYGEN: MIT Project Oxygen – Pervasive Human-Centered Computing. Disponível em: <http://oxygen.lcs.mit.edu/>. Acesso em: jan. 2005.

ROMÁN, M. et. al. Gaia: A Middleware Infrastructure to Enable Active Spaces. **IEEE Pervasive Computing**, [S.l.], p. 74-83, Oct./Dec. 2002.

SATYANARAYANAN, M. Pervasive Computing: Vision and Challenges. **IEEE Personal Comunications**, New York, v.4, n.8, Aug. 2001. Disponível em: <http://ieeexplore.ieee.org/xpl/abs_free.jsp?arNumber=943998>. Acesso em: ago. 2005.

SILVA, L. **Primitivas para suporte à distribuição de objetos direcionados à pervasive computing**. 2003. 123 p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SCHAEFFER FILHO, A. **PerDiS**: um serviço para descoberta de recursos no ISAM pervasive environment. 2005. 103 p. Dissertação (Mestrado em Ciência da Computação) – Instituto de Informática, UFRGS, Porto Alegre.

SCHAEFFER FILHO, A.; SILVA, L.; YAMIN, A.; AUGUSTIN, I.; MORAIS, L.; GEYER, C. PerDiS: A Scalable Resource Discovery Service for the ISAM Pervasive Environment. In: INTERNATIONAL WORKSHOP ON HOT TOPICS IN PEER-TO-PEER SYSTEMS, 2004. **Proceedings…** Los Alamitos: IEEE Computer Society Press, 2004.

SCHMIDT, A.; BEIGL M.; GELLERSEN, H. There is more to context than location. In: WORKSHOP ON INTERACTIVE APPLICATIONS OF MOBILE COMPUTING, IMC, 1998, Rostock, Germany. **Proceedings…** Rostock: Never Hochschulschriftenverlag, 1999.

SCHMIDT, A. Advanced interaction in context. In: INTERNATIONAL SYMPOSIUM ON HANDHELD AND UBIQUITOUS COMPUTING, HUC, 1., 1999. **Proceedings…** Berlin: Springer-Verlag, 1999.

YAMIN, A. **Arquitetura para um ambiente de Grade Computacional Direcionado à Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva.** 2004. 195 f. Tese (Doutorado em Ciência da Computação)- Instituto de Informática, UFRGRS, Porto Alegre.

WEISER, M. The computer of the 21st Century. **Scientific American**, [S.l.], v.265, n.3, p.66–75, Sept. 1991.

WHITE, A. **Performance and interoperability in Solar**. 2002. (Technical Report TR2002-427). Disponível em: <ftp://ftp.cs.dartmouth.edu/TR/TR2002-427.pdf> Acesso em: jun. 2004.

WORLDNET a lexical database for the english language. Disponível em: <http://wordnet.princeton.edu/>. Acesso em: dez. 2004.

WSAD: WebSphere Studio Application Developer. Disponível em: <http://www-306.ibm.com/software/integration/wsadie/>. Acesso em: jul. 2006.

ZIMMER, T. Towards a Better Understanding of Context Attributes. In: IEEE ANNUAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS, 2., 2004. **Proceedings...** Los Alamitos: IEEE, 2004.

APÊNDICE A VERIFICAÇÃO DE CICLOS NO GRAFO DE INFERÊNCIA DE CONTEXTO

A transformação das árvores de inferência de contexto em grafos tem por objetivo aumentar o desempenho do servidor de contexto, possibilitando que mais aplicações sejam atendidas em uma dada instância. Essa modificação, entretanto, deve ser refletida no procedimento de cálculo do valor dos dados de contexto. Busca-se através deste anexo, demonstrar que os grafos de contexto gerados pelo MultiS não possuem ciclos, e portanto, quando um elemento de contexto é modificado, basta que sejam notificados os outros nodos que utilizam sua informação, sem a necessidade de verificação de existência de ciclos entre os nodos envolvidos nesse processo.

A definição de um elemento de contexto pode ser entendida como o trecho CD-XML que contém sua descrição. Ela inclui todos os filtros, sensores e demais elementos de contexto utilizados para produzir seu valor. O módulo `Verifier`, antes de instanciar um novo elemento de contexto, consulta a definição dos elementos de contexto já existentes. Caso seja encontrado algum elemento com a mesma definição CD-XML, não se adiciona um novo objeto ao servidor de contexto. Ao invés disso, acrescenta-se uma referência a esse objeto já existente. Essa operação é *thread safe*, e portanto, não pode existir em uma instância MultiS dois elementos com a mesma definição CD-XML. Essa verificação é feita pelo módulo Verifier, que armazena as definições CD-XML dos elementos instanciados.

A definição CD-XML de um elemento contém todos os objetos que são utilizados por ele. Portanto, se um objeto A é utilizado por um objeto B, a definição de A está contida no trecho CD-XML que define o elemento B.

Para que existisse um ciclo no servidor de contexto, deveriam existir dois objetos X, Y de tal forma que X utilizasse Y e Y utilizasse X. Se isso é verdade, então a definição de X deveria estar contida na definição Y, e vice-versa. Eles teriam, portanto, a mesma definição CD-XML. A conferência feita pelo módulo `Verifier`, entretanto, garante a inexistência de mais de um objeto de contexto com a mesma definição CD-XML. É possível afirmar-se, então, que não existem ciclos nos grafos de inferência de contexto.

APÊNDICE B RESULTADOS DOS TESTES

A Tabela B.1 apresenta os dados relativos aos testes de execução da aplicação PerMuseum sem o compartilhamento de sub-árvores de contexto. Os valores são apresentados em milisegundos e são relativos ao tempo necessário para o processamento de todas as árvores de inferência de contexto após uma modificação no ambiente. Destaca-se que o tempo apresentado é relativo apenas ao processamento das informações (*ie.* não considera-se o tempo de detecção da alteração pelo sistema de monitoramento). Os valores apresentados foram obtidos através da primitiva `System.currentTimeMillis()` da linguagem Java.

Tabela B.1: Testes de execução sem compartilhamento de sub-árvores.

Execução /Aplicações simultâneas	1	5	10	20	50	100
1	26	42	153	357	247	2226
2	9	21	85	136	278	362
3	17	35	72	83	762	287
4	13	122	86	96	112	383
5	8	11	66	76	126	2528
6	6	71	55	62	216	225
7	5	56	42	98	202	278
8	20	10	45	613	191	2019
9	2	56	54	187	182	2031
10	18	55	135	150	494	304
11	11	37	77	207	303	1904
12	18	24	176	71	195	532
13	10	29	97	108	138	564
14	13	68	79	485	172	439
15	7	137	77	68	204	476
16	12	86	108	211	190	1901
17	12	30	48	150	146	252
18	10	6	20	184	427	436
19	14	52	69	135	244	1756
20	23	39	52	112	226	253

A Tabela B.2 apresenta os dados relativos aos testes de execução da aplicação PerMusem com o compartilhamento de informações de contexto. Os testes das tabelas B.1 e B.2 foram executados nas mesmas máquinas e sob as mesmas circunstâncias.

Tabela B.2: Testes de execução com o compartilhamento de sub-árvores.

Execução /Aplicações simultâneas	1	5	10	20	50	100
1	16	20	33	139	231	464
2	1	68	32	173	182	160
3	33	16	20	221	134	153
4	10	48	27	81	185	160
5	1	81	29	87	141	235
6	12	33	79	157	125	210
7	18	24	46	50	104	451
8	22	98	12	41	98	158
9	1	2	43	109	139	171
10	2	17	13	94	67	410
11	1	4	12	126	51	134
12	2	24	150	46	154	350
13	14	28	24	132	155	225
14	18	15	96	29	251	154
15	2	81	61	42	164	529
16	2	36	14	163	184	379
17	19	56	33	38	102	161
18	23	82	57	142	58	217
19	12	12	92	147	85	218
20	14	12	18	117	136	158