UNIVERSIDADE CATÓLICA DE PELOTAS
CENTRO POLITÉCNICO
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Revisando a Sensibilidade ao Contexto na Computação Ubíqua

por
Luthiano Venecian

Trabalho Individual I
TI-2008-2-06

Orientador: Prof. Dr. Adenauer Corrêa Yamin

Pelotas, dezembro de 2008

AGRADECIMENTOS

Agradeço a Deus e aos meus pais pelo carinho e apoio constante em cada momento da minha vida.

A meu orientador, Prof. Dr. Adenauer Yamin, pela sua orientação, amizade e motivação durante esse período de convivência, que em muitas situações foram essências para minha permanência nesse curso de mestrado.

Aos meus colegas, em especial a Nelsi, que além de ser uma pessoa talentosa, é uma amiga presente em todos os momentos desta jornada.

Ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pelo apoio financeiro.

Ao Escritório Administrativo da Renovação Carismática Católica do Brasil pela confiança depositada.

SUMÁRIO

LISTA DE FIGURAS 5

LISTA DE TABELAS 6

LISTA DE ABREVIATURAS E SIGLAS 7

RESUMO 9

ABSTRACT 10

1 INTRODUÇÃO 11
1.1 Tema 12
1.2 Motivação 12
1.3 Objetivos 13
1.4 Estrutura do Texto 13

2 COMPUTAÇÃO UBÍQUA 14
2.1 Características 14
2.1.1 Acesso Ubíquo 15
2.1.2 Comportamento Inteligente 16
2.1.3 Interação Natural 16
2.2 Redes de Comunicação Ubíquas 18
2.2.1 Mobilidade 18
2.2.2 Descoberta de Nós 19
2.3 Considerações Sobre o Capítulo 20

3 SENSIBILIDADE AO CONTEXTO 21
3.1 Definindo Contexto 21
3.2 Características da Informações Contextuais 22
3.3 Dimensões de Informação de Contexto 23
3.4 Modelagem de Contexto 25
3.5 Aquisição de Contexto 27
3.6 Interpretação de Contexto 27
3.7 Armazenamento de Informações Contextuais 28
3.8 Arquitetura de Aplicações Sensíveis ao Contexto 28
3.9 Uso de Aplicações Sensíveis ao Contexto 29
3.10 Considerações Sobre o Capítulo 29

4 ONTOLOGIAS: CONCEITOS E TECNOLOGIAS 31
4.1 O Conceito de Ontologia 31
4.2 Tipos de Ontologias 32
4.3 Benefícios das Ontologias 33
4.4 Linguagens para Ontologias 34
4.4.1 RDF 34
4.4.2 RDF Shema 35
4.4.3 OWL 35
4.5 Considerações Sobre o Capítulo 36

5 PLATAFORMAS SENSÍVEIS AO CONTEXTO 37
5.1 Context Kernel 37
5.1.1 Arquitetura 38
5.2 Context Toolkit 40
5.2.1 Arquitetura 40
5.3 Infraware 42
5.3.1 Arquitetura 42
5.4 Moca 44
5.4.1 Arquitetura 44
5.5 Tabela Comparativa 46
5.6 Considerações Sobre o Capítulo 46

6 CONSIDERAÇÕES FINAIS 48

REFERÊNCIAS 50

LISTA DE FIGURAS

Figura 2.1 Características de um Ambiente Ubíquo (HARIHAR K.; KURKOVSKY, 2005) 15
Figura 2.2 Diferentes Técnicas para Interação com Objetos Físicos. Extraído de (BROLL, 2007) 17
Figura 2.3 Abordagens para Descoberta de Nós (HARIHAR K.; KURKOVSKY, 2005) 20

Figura 4.1 A Ontolíngua e os Formalismos para os quais podem ser Traduzidas as Ontologias (A. FARQUHAR R. FIKES, 1996) 34

Figura 5.1 Fluxo de Informação do Context Kernel (ARRUDA JR., 2003) 38
Figura 5.2 Componentes do Context Toolkit (DEY, 2000) 41
Figura 5.3 Plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006) . . . 43
Figura 5.4 Serviços da Arquitetura Moca (SACRAMENTO et al., 2004) 45

LISTA DE TABELAS

Tabela 3.1 Avaliação das abordagens para Modelagem de Contexto 27

Tabela 5.1 Comparação entre Plataformas Sensíveis ao Contexto 46

LISTA DE ABREVIATURAS E SIGLAS

API	Application Programming Interface
CML	Context Modelling Language
DOM	Document Object Model
EXEHDA	Execution Environment for Highly Distributed Applications
Foaf	Friend of a Friend
GSM	Global System for Mobile Communications
GTSH	Gator Tech Smart House
HCI	Human Computer Interaction
HTML	Hiper Text Markup Language
ISAM	Infra-estrutura de Suporte às Aplicações Móveis Distribuídas
OML	Ontology Markup Language
OPEN	Ontology-Driven Pervasive Environment
ORM	Object-Role Modeling
OSGi	Open Service Gateway Initiative
OWL	Web Ontology Language
PDA	Personal Digital Assistant
RDF	Resource Description Language
RFID	Radio Frequency Identification
SQL	Structured Query Language
SO	Sistema Operacional
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SWS	Serviços Web Semânticos
UML	Unified Modeling Language
URI	Universal Resource Identifier
WASP	Web Architectures for Services Platforms

W3C	World Wide Web Consortium
XML	Extensible Markup Language
XOL	Ontology Exchange Language

RESUMO

Este Trabalho Individual tem como objeto o estudo de conceitos relacionados a Computação Ubíqua, Sensível ao Contexto, Ontologias, culminando com um estudo comparatório entre Plataformas Sensíveis ao Contexto. Com os avanços tecnológicos temos dispositivos menores e com maior poder de computação e comunicação. Um Ambiente Ubíquo contém diferentes dispositivos, tais como sensores, atuadores, eletroeletrônicos e dispositivos móveis que interagem com a pessoa de forma natural ao conhecer o contexto. A diversidade de dispositivos e informações do Ambiente Ubíquo introduz um problema de interoperabilidade. Um Ambiente Ubíquio é dinâmico devido à mobilidade do usuário e a grande variedade de dispositivos existentes. Portanto, ao se construir e executar aplicações ubíquas sensíveis ao contexto, há uma série de funcionalidades que devem ser providas, envolvendo desde a aquisição de informações contextuais, a partir do conjunto de fontes heterogêneas e distribuídas, até a representação dessas informações, seu processamento e armazenamento.

Palavras-chave: Computação Ubíqua, Sensibilidade ao Contexto, Ontologias.

TITLE: “REVIEWING THE RECALL IN THE CONTEXT UBIQUITOUS COMPUTING”

ABSTRACT

This work has the single object of the study concepts related to Ubiquitous Computing, the Context Sensitive, Ontologies, culminating with a study comparatório between the Context Sensitive platforms. With technological advances we have devices smaller and more power to computing and communication. A ubiquitous contains various devices such as sensors, actuators, electronics and mobile devices that interact with the person in a natural way to understand the context. The diversity of devices and information of ubiquitous computing introduces a problem of interoperability. How is a dynamic environment due to the mobility of the user and the variety of existing devices. So when you build and run applications ubiquitous sensitive to the context, there is a number of features that should be provided, since involving the acquisition of contextual information from the set of heterogeneous and distributed sources, by the representation of such information, processing and storage.

Keywords: Ubiquitous Computing , Context-aware, Ontologies.

1 INTRODUÇÃO

Este capítulo apresenta o tema do Trabalho Individual, destacando as motivacões e objetivos, bem como descreve a estrutura do texto como um todo.

A Computação Ubíqua (SATYANARAYANAN, 2001), é uma forma de computação onde o processamento está espalhado no ambiente através de vários dispositivos, que executam tarefas bem definidas dependendo de sua natureza, interligados de forma que essa estrutura torna-se invisível para o usuário. Aplicações ubíquas executam em ambientes instrumentados com sensores, geralmente dotados de interfaces de redes sem fio, nos quais dispositivos, agentes de software e serviços são integrados de forma transparente e cooperam para atender aos objetivos da aplicação. Essa categoria de aplicações caracteriza-se por constantes mudanças em seu estado de execução, geradas pelos ambientes altamente dinâmicos em que executam.

A Sensibilidade ao Contexto refere-se à capacidade de uma aplicação de perceber características de seu ambiente, e é um requisito chave para permitir a adaptação em resposta às mudanças ambientais. Aplicações sensíveis ao contexto, onde contexto é ”qualquer informação que pode ser usada para caracterizar a situação de uma entidade (pessoa, local ou objeto) que é considerada relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e a aplicação (DEY, 2000).”, são aplicações que conhecem o ambiente no qual estão sendo utilizadas e tomam decisões de acordo com mudanças no seu próprio ambiente. Ou seja, reagem a ações executadas por outras entidades, podendo essas ser pessoas, objetos ou até mesmo outros sistemas, que modifiquem o ambiente. Essas aplicações, de um modo geral, utilizam sensores para tomar ciência de modificações que venham a acontecer no ambiente. Tais modificações são alterações nas informações de contexto.

Com o avanço recente da computação móvel, a computação ubíqua pode fazer uso de dispositivos móveis para que sistemas estejam cada vez mais centrados nos usuários, cientes das freqüentes variações das informações de contexto que são inerentes a esses sistemas. Como exemplo de dispositivos móveis podemos citar os handhelds e smartphones que, além de prover cada vez mais um maior poder computacional, utilizam redes sem fio para se comunicarem com outros dispositivos ou com a Internet.

Um ambiente ubíquo tem uma natureza dinâmica, devido à mobilidade do usuário, a variedade de dispositivos e tecnologias existentes, assim como às mudanças constantes nos perfis dos usuários (ZHOU Y.; CAO, 2007). Para fornecer suporte ao dinamismo do ambiente ubíquo, requer a definição das suas regras de comportamento em tempo de execução (WALTENEGUS, 2006).

A modelagem de contexto utilizando ontologias permite a definição do comporta-

mento do ambiente ubíquo em tempo de execução, mas não fornece o suporte necessário para lidar com o dinamismo do ambiente. Uma das alternativas para lidar com o dinamismo é o uso de Servicos Web Semânticos.

Portanto, ao se construir e executar aplicações ubíquas sensíveis ao contexto, há uma série de funcionalidades que devem ser providas, envolvendo desde a aquisição de informações contextuais, a partir do conjunto de fontes heterogêneas e distribuídas, até a representação dessas informações, seu processamento, armazenamento, e a realização de inferências para seu uso em tomadas de decisão. Em vez de deixar essas funcionalidades a cargo da aplicação, as incorporando ao código do negócio, são utilizadas como infra-estruturas subjacentes as plataformas ou middlewares de provisão de contexto (ABOWD G. D.; RODDEN, 2002).

1.1 Tema

Este Trabalho Individual tem como enfoque principal o estudo de plataformas sensíveis ao contexto.

A computação ubíqua tem como requisito a manipulação de diferentes contextos de execução. Um dos aspectos deste tema, a sensibilidade ao contexto, é considerada um dos grandes desafios desta área de pesquisa.

Assim, o tema deste trabalho abrange estudos que visam comparar diferentes plataformas sensíveis ao contexto, através do estabelecimento da relação existente entre Computação Ubíqua, Sensível ao Contexto e o emprego de Ontologias nessas plataformas estudadas.

1.2 Motivação

Aplicações ubíquas são tipicamente sensíveis ao contexto no qual o usuário encontra-se. Essas aplicações recebem dados de sensores, de dispositivos e de ações de usuários e provêem um alto suporte a mobilidade (SATYANARAYANAN, 2001). Nessas aplicações, informações de contexto relativas ao usuário, ambiente e localização tendem a mudar com freqüência e, consequentemente, eventos contextuais emergem de forma concorrente e dinâmica, fazendo-se necessário o uso da arquitetura baseada em eventos.

(CHEN G., 2000) afirma que ”o serviço de contexto é responsável por entregar mudanças de contexto aos clientes que se subscreveram para as mudanças de contexto relacionadas”. A concorrência e dinamicidade de eventos contextuais podem ser exemplificados com o estudo de caso apresentado em (AL., 2006) onde um paciente acometido com uma séria doença cardíaca pode ter suas funções cardíacas monitoradas através de sensores e, caso seja identificado um estado preocupante, pessoas da família, seu médico e até mesmo uma ambulância podem ser notificados. Um exemplo de uma situação preocupante seria caso os sensores identificassem que a pressão sanguínea e a quantidade de batimentos por minuto de seu coração encontram-se em uma faixa perigosa. Desse modo, mecanismos de comunicação baseada em eventos que provêem suporte para composição de eventos concorrentes permitem uma maior expressividade na declaração de interesses. Tipicamente, sistemas que dão suporte a composição de eventos utilizam como infra-estrutura subjacente algum sistema publish/subscribe que trabalha com eventos primitivos.

Portanto, a principal motivação para este Trabalho Individual, é o estudo de plataformas sensíveis ao contexto com base nas características contextuais seníveis ao contexto.

1.3 Objetivos

O objetivo geral deste Trabalho Individual é explorar a correlação entre computação ubíqua, sensível ao contexto e ontologias, avaliando o estudo das plataformas sensíveis ao contexto.

Os objetivos específicos são:

- caracterizar as plataformas sensíveis ao contexto em ambientes de execução para computação ubíqua;
- estudar os fundamentos teóricos sobre computação ubíqua, computação sensível ao contexto e ontologias;
- compreender a correlação entre computação ubíqua, sensível ao contexto e ontologias;
- estudar métodos para manipulação de ontologias;

1.4 Estrutura do Texto

A estrutura do texto deste Trabalho Individual é a seguinte:

- Capítulo 2: Computação Ubíqua: este capítulo apresenta fundamentos teóricos da computação ubíqua;
- Capítulo 3: Sensibilidade ao Contexto: são conceitualizados os principais conceitos da computação sensível ao contexto e descritos cenários relacionados à essa área;
- Capítulo 4: Ontologias: apresenta os fundamentos teóricos relacionados com ontologias, destacando linguagens para manipulação de ontologias;
- Capítulo 5: Plataformas Sensíveis ao Contexto: são apresentadas plataformas relacionadas a sensibilidade ao contexto e suas interligações;
- Capítulo 6: Considerações Finais.

2 COMPUTAÇÃO UBÍQUA

A visão da Computação Ubíqua endereça o aumento do número de dispositivos e tecnologias em nosso ambiente natural. Segundo esta visão (WEISER, 1995) os recursos de computação serão onipresentes na vida diária e estarão interconectados com a finalidade de fornecer informação e/ou serviços aos usuários em qualquer lugar e momento.

A Computação Ubíqua proporciona uma interação natural entre a pessoa e o ambiente, a qual requer uma mínima intervenção humana e acontece de forma autônoma, interativa e relevante (SATYANARAYANAN, 2001). Este tipo de interação entre as pessoas e seu ambiente é um dois maiores desafios da Computação Ubíqua, cujo interesse é de adequar-se a tecnologia a nossas vidas diárias (WEISER, 1995).

A Computação Ubíqua tem o potencial de mudar a forma como desenvolvemos as nossas atividades cotidianas (HARIHAR K.; KURKOVSKY, 2005). Desse modo em um Ambiente Ubíquo não se percebe que as pessoas estão interagindo com as máquinas.

2.1 Características

Segundo (ABOWD G. D.; RODDEN, 2002), a Computação Ubíqua promete mais do que uma infra-estrutura, mas acima de tudo, novos paradigmas de interação inspirados pelo amplo acesso à informação e às capacidades computacionais. Os autores assinalam três desafios ou características nos Ambientes Ubíquos, sendo eles: i) interfaces naturais; ii) ciente de contexto; e iii) captura e acesso automatizado.

Outras divisões de características dos Ambientes Pervasivos são dadas por (HARIHAR K.; KURKOVSKY, 2005):

- acesso Ubíquo;
- comportamento inteligente;
- interação natural;
- ciente de contexto.

Os autores apontam também que além dessas características, os Ambientes Ubíquos devem ser confiáveis e seguros.

Na figura 2.1, são apresentadas as quatro características de um Ambiente Ubíquo propostas por (HARIHAR K.; KURKOVSKY, 2005).

A seguir, serão detalhadas essas características, sendo que a característica ciente de contexto será apresentada com maior detalhe na Seção 3 por ser o foco deste trabalho.

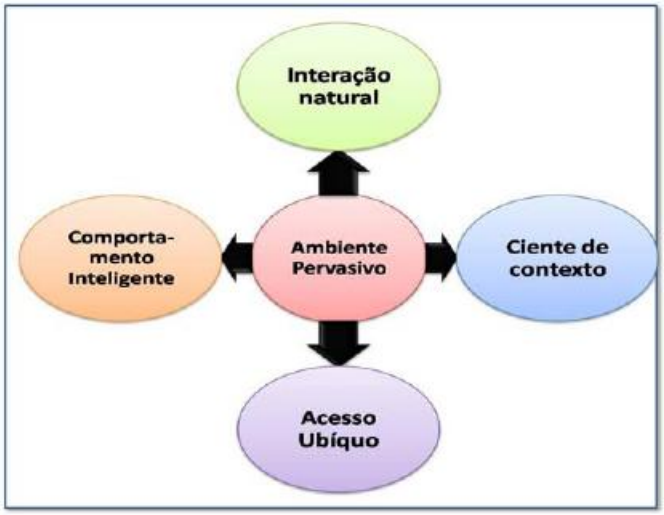


Figura 2.1: Características de um Ambiente Ubíquo (HARIHAR K.; KURKOVSKY, 2005)

2.1.1 Acesso Ubíquo

Acesso ubíquo refere-se ao acesso à informação e aos serviços do ambiente em qualquer lugar e momento. Segundo (HARIHAR K.; KURKOVSKY, 2005), uma diferença importante entre a Computação Tradicional e a Computação Ubíqua está no acesso à informação e aos serviços. No paradigma tradicional, o usuário realiza manualmente algumas tarefas, mas na Computação Ubíqua, o usuário acessa de forma mais natural ou transparente (GRIMM R.; DAVIS J.; LEMAR, 2004).

Devido ao dinamismo do Ambiente Ubíquo, é necessária uma tecnologia robusta que possa integrar com os dispositivos de uma maneira uniforme (HARIHAR K.; KURKOVSKY, 2005). Como exemplo, temos os protocolos de localização dinâmica de serviços (EDWARDS, 2006).

A seguir, são descritas algumas arquiteturas middleware utilizadas para fornecer acesso ubíquo como: Jini (JINI, 2008), Open Service Gateway Initiative (OSGi) (OSGI, 2008) e EXEHDA (YAMIN, 2004).

O Jini (JINI, 2008), é um protocolo de localização dinâmica de serviços que pode ser utilizado para a construção de sistemas de redes adaptativas, sendo elas: escaláveis, adaptáveis e flexíveis. Essas características são requeridas em ambientes de computação dinâmicos.

Harihar e Kurkovsky (HARIHAR K.; KURKOVSKY, 2005) discutem o uso da tecnologia de redes Jini para possibilitar e integrar tecnologias usadas por Ambientes Ubíquos. Já Edwards (EDWARDS, 2006) realiza uma comparação do Jini e outros protocolos de localização dinâmica de serviços em Ambientes Ubíquos

Outro *middleware* para Ambientes Ubíquos é o OSGI, o qual foi construído em 1999 e define uma especificação livre para o envio e o fornecimentos de múltiplos serviços em residências, automóveis e em outros ambientes.

O Gator Tech Smart House (GTSH) (HELAL, 2005) é um projeto da Universidade da Flórida cujo objetivo é de criar Ambientes Ubíquos de última geração apresentando um arquitetura *middleware* aplicável para qualquer Ambientes Ubíquo. A arquitetura proposta do GTSH para desenvolver Ambientes Ubíquos utiliza *middleware* OSGi e a

modelagem de contexto com ontologias.

O EXEHDA é um *middleware* adaptativo ao contexto e baseado em serviços que visa criar e gerenciar um ambiente ubíquo, bem como promover a execução, sob este ambiente, das aplicações que expressam a semântica siga-me. Estas aplicações são distribuídas, móveis e adaptativas ao contexto em que seu processamento ocorre, estando disponíveis a partir de qualquer lugar, todo o tempo. O *middleware* faz parte dos esforços de pesquisa do Projeto ISAM. O ISAM vem sendo desenvolvido por um consórcio de universidades gaúchas, e foi iniciado na UFRGS.

2.1.2 Comportamento Inteligente

A característica comportamento ”Inteligente”de um Ambiente Ubíquo, refere-se à sua habilidade de se adaptar ao comportamento do usuário de uma forma personalizada para suprir aos usuários informação no tempo e lugar correto (HARIHAR K.; KURKOVSKY, 2005).

Em seguida são citados trabalhos que utilizam aprendizagem de máquina, agentes de software e SWS, como técnicas que possibilitam um comportamento ”inteligente”em Ambientes Ubíquos.

(KRAUSE A.; SMAILAGIC, 2006), utilizam técnicas de aprendizagem de máquina e análise estatística em um dispositivo móvel ciente de contexto. Os autores desenvolveram um protótipo SenSay que utiliza vários sensores vestíveis (wearables) para obter a atividade e estado psicológico da pessoa.

(J., 2006) descrevem um método visual para a animação dos movimentos das mãos gerados por acelerômetros. Os acelerômetros produzem séries de tempo executadas pelos movimentos das mãos. Os autores utilizam Hidden Markov Models (HMM) para a modelagem de tais séries de tempo. HMM são máquinas de estados finitos capazes de gerar e analisar uma série de tempo mediante um modelo probabilístico.

(ZHOU Y.; CAO, 2007) apresentam uma arquitetura baseada em agentes com o nome MDAgent. O MDAgent utiliza duas funcionalidades dos agentes de software: autonomia e mobilidade. O agente autônomo é responsável pelo raciocínio e a tomada de decisão de acordo com a informação de contexto, enquanto que o agente móvel é responsável pela migração de componentes de aplicação de acordo com a nova localização da pessoa. O MDAgent utiliza ontologias na modelagem de contexto.

A aplicação dos Serviços Web Semânticos possibilita uma nova faixa de tarefas de automação, as quais são executadas por pessoas, como a descoberta automatizada, invocação e composição de serviços (MCILRAITH S.; MARTIN, 2003).

(BROENS, 2004) utilizam SWS e informação de contexto. Esse trabalho focase na descrição do algoritmo matching que utiliza atributos contextuais. (MOKHTAR S.; FOURNIER, 2005) apresentam uma proposta para a composição de serviços cientes de contexto em Ambientes Pervasivos. Os autores modelam serviços e tarefas do usuário em OWL-S e as complementam com informação de contexto.

2.1.3 Interação Natural

A interação natural em Ambientes Ubíquos refere-se a uma interação homem-computador (Human Computer Interaction) (HCI) de forma quase ou se possível natural. Interagimos com o mundo real com as mãos, os gestos, as palavras e com o mundo digital usando interfaces com o uso do mouse, teclado, entre outros.

As interfaces de computador que suportam mais de uma forma de comunicação natural humana (voz, gestos, handwriting) estão substituindo elementos do paradigma da interação Graphical User Interface (GUI) que majoritariamente utiliza o teclado e o mouse.

Um desafio na comunidade científica de Computação Ubíqua é a necessidade de gerenciar complexas interações entre numerosos dispositivos interconectados (HELAL, 2005). Para fornecer uma interação natural em um Ambiente Pervasivo é requerida a investigação de novas metáforas de interação HCI (J., 2006). A seguir apresentam-se trabalhos que utilizam diferentes metáforas de interação HCI para fornecer uma interação natural.

(BROLL, 2007) apresentam diferentes formas de interação de um dispositivo móvel com objetos físicos (physical mobile interactions). Os autores aumentaram as funcionalidades dos dispositivos móveis para a interação dos objetos do cotidiano. A Figura 2.2 apresenta diferentes formas de interação de um dispositivo móvel com objetos físicos:

- pelo toque de objetos físicos (*touching*)(Figura 2.2 - a), os objetos físicos são aumentados com tags RFIDs, os quais armazenam informação. Os usuários selecionam esses objetos pelo toque e lêem que contem;
- pela sinalização dos objetos físicos (*pointing*)(Figura 2.2 - b), os objetos físicos são aumentados com marcadores visuais que podem ser capturados com câmeras de telefones moveis e interpretados por algoritmos de reconhecimento de imagens;
- localização (Figura 2.2 - c), uso de dispositivos GPS externos via Bluetoth para medir a proximidade de objetos físicos e interagir com eles apropriadamente;
- escanear (Figura 2.2 - d), usa as funcionalidades Bluethot dos celulares de procurar, conectar-se e interagir com outros dispositivos e objetos aumentados com sua visinhança.

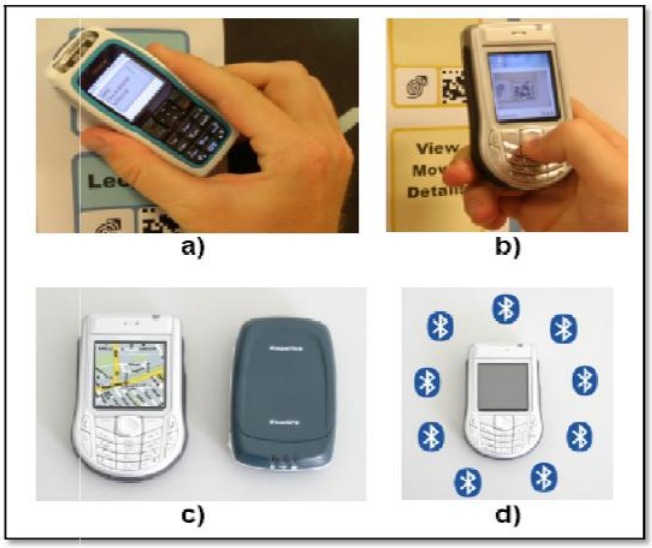


Figura 2.2: Diferentes Técnicas para Interação com Objetos Físicos. Extraído de (BROLL, 2007)

2.2 Redes de Comunicação Ubíquas

No âmbito das redes de comunicação ubíquas, o foco está nos detalhes envolvidos na comunicação entre dispositivos em ambientes ubíquos. Portanto, estudos nessa área variam desde a criação de interfaces de rede sem fio até o desenvolvimento de protocolos como os de transporte, roteamento e mobilidade. Nesta seção serão apresentados *Mobilidade* e *Descoberta de nós*, elementos fundamentais para a criação de ambientes ubíquos.

2.2.1 Mobilidade

O conceito de mobilidade está relacionado à possibilidade de um nó migrar entre diferentes redes, e ainda assim manter as conexões estabelecidas com os nós da rede de origem. Para um melhor entendimento, imagine uma pessoa que esteja de mudança. Nesse processo, dentre várias outras preocupações, essa pessoa deve querer certamente alterar os endereços de entrega de correspondências como revistas e faturas de cartão de crédito. Sendo assim, a mesma deverá então entrar em contato com as editoras das revistas e a companhia do seu cartão de crédito, para notificá-las que seu endereço mudou, e que portanto, as correspondências devem ser entregues no novo endereço. Do contrário, suas revistas e faturas de cartão de crédito serão enviadas ao endereço antigo. De uma maneira geral, isso é o que acontece em cenários caracterizados pela mobilidade de seus nós. Quer dizer, ao se mover de uma rede para outra, um nó deve disponibilizar seu novo endereço, na rede de destino, àqueles com os quais possui conexões estabelecidas. Dessa forma, conexões pendentes com outros nós podem ser restabelecidas. Essa possibilidade de disponibilizar comunicação aos dispositivos, mesmo em movimento, permite às aplicações trabalhar em segundo plano, procurando invisivelmente por serviços de interesse dos usuários, à medida que estes se deslocam entre diferentes ambientes. No entanto, aplicações desse tipo encontram-se diante de um novo conjunto de problemas, os quais podem ser agrupados da seguinte forma (SATYANARAYANAN, 2001).

1. Limitação de recursos: é fato que os dispositivos móveis são limitados em termos de recursos, quando comparados com computadores pessoais. A velocidade dos processadores e a capacidade de memória e disco são consideravelmente maiores nestes últimos do que nos dispositivos móveis.

2. Restrições de energia: enquanto que os computadores pessoais são ligados a uma rede elétrica, os dispositivos móveis, diferentemente, dependem de baterias. Sendo as mesmas de capacidade limitada, as aplicações móveis devem preferivelmente contemplar técnicas para economizar energia.

3. Variabilidade dos enlaces sem fio: a qualidade das redes sem fio ainda é bastante variável, tanto em termos de desempenho quando de confiabilidade. Enquanto alguns ambientes disponibilizam conexões confiáveis e com razoável largura de banda, em outros isso não acontece. Essa variabilidade torna-se ainda mais evidente em ambientes abertos, nos quais o enlace sem fio pode ser compartilhado por diversos usuários em certos momentos, enquanto em outros por um número bastante reduzido dos mesmos.

4. Segurança: devido à natureza de difusão (i.e., broadcast) dos enlaces sem fio, torna-se mais fácil interceptar mensagens nos mesmos do que nos enlaces cabeados. Por-

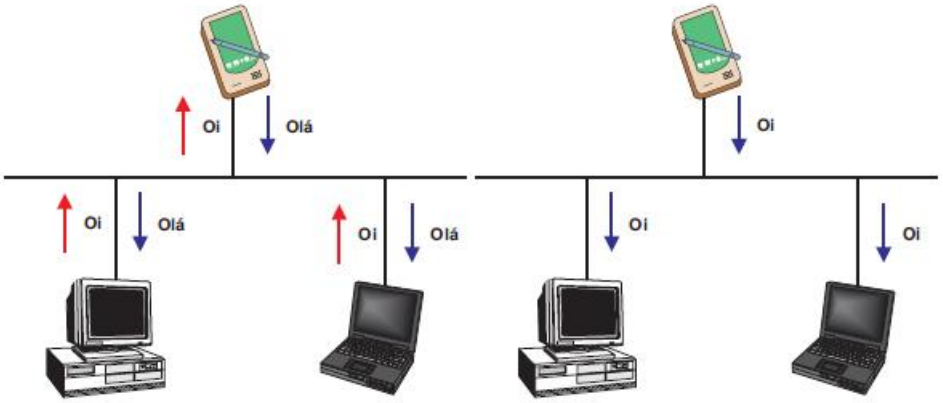
tanto, se segurança já é um aspecto importante nestes últimos, em enlaces sem fio isso ainda é mais crítico.

A mobilidade, no entanto, além gerar os problemas descritos, tem um efeito direto sobre a estrutura da rede. Dessa forma, a flexibilidade da mesma deve ser proporcional à mobilidade de seus nós. Em redes ethernets cabeadas, por exemplo, os nós são estáticos. Portanto, apenas em situações esporádicas torna-se necessário alterar o endereço de rede de um nó. Nesse caso, protocolos como DHCP resolvem transparentemente o problema de reconfiguração de endereço dos nós. No outro extremo estão as redes populadas por nós completamente móveis. Esse nível de mobilidade permite aos usuários se moverem para áreas que não possuem cobertura de rede. Em cenários desse tipo, redes ponto a ponto sem infra-estrutura física fixa, conhecidas como redes ad hoc (CHLAMTAC I., 2003), são mais apropriadas. Ou seja, os nós deveriam ser capazes de estabelecer conexões diretamente uns com os outros, sempre que preciso, sem depender de nenhuma infra-estrutura física. Nesse escopo, como (SUN J. Z., 2002) já afirmaram, três modos de comunicação podem ser diferenciados, com relação ao grau de mobilidade: nomádico, celular e pervasivo. No primeiro modo nenhuma comunicação é necessária quando um nó está migrando de uma rede para outra. Um exemplo típico da comunicação nomádica é quando um usuário utiliza um computador portátil (i.e., um notebook) para se conectar a uma rede em seu trabalho e outra em sua casa. Perceba que, não há necessidade de manter as conexões de rede enquanto o usuário está se movendo do trabalho para sua casa. Já no modo de comunicação celular, a rede é organizada em células, adjacentes umas às outras. Além disso, cada célula tem um elemento central, o qual provê conectividade para todos os nós na mesma. Um nó pode, portanto, mover-se entre diferentes células e, mantendo contato com seus respectivos centralizadores, tornar-se acessível aos outros nós. As atuais redes de telefonia móvel são um exemplo desse modo de comunicação, nas quais as Estações Rádio Base (ERBs) atuam como centralizadoras. Por fim, o modo de comunicação pervasivo pode ser caracterizado tanto pela descentralização como pela falta de infra-estrutura de rede fixa, diferentemente dos outros dois modos. A rede é, portanto, formada espontaneamente, à medida que mais e mais nós se aglomeram em uma determinada área. Dentre as atuais soluções no contexto de mobilidade podemos citar Mobile IP (PERKINS, 1997), GPRS e *Bluetooth*. Basicamente, os dois primeiros são soluções de mobilidade para redes IP e de telefonia móvel, respectivamente. *Bluetooth*, por outro lado, é uma tecnologia de baixo consumo de energia para a criação de redes *ad hoc* de curto alcance.

2.2.2 Descoberta de Nós

Colocando de maneira simples, a descoberta de nós está associada à capacidade de um nó em descobrir outros na rede, e de ser descoberto pelos mesmos. De uma maneira geral, esse processo pode ser realizado de duas formas: através de consultas e através de notificações. Na primeira, um nó envia mensagens de consulta de forma a descobrir os nós presentes na rede. Ao recebê-las, os mesmos podem então enviar uma mensagem de resposta ao requisitante, para informá-lo de sua presença na rede. Esse cenário é ilustrado na Figura 2.3(a). No outro modo de descoberta, ilustrado na Figura 2.3(b), um nó envia mensagens periodicamente para notificar aos demais que o mesmo está presente na rede. Note, portanto, que nessa abordagem nenhuma mensagem de resposta é necessária.

Esse conceito, apesar de simples, é bastante útil em ambientes de computação per-



(a) Descoberta de nós utilizando *queries*. (b) Descoberta de nós utilizando notificações.

Figura 2.3: Abordagens para Descoberta de Nós (HARIHAR K.; KURKOVSKY, 2005)

vasiva, principalmente naqueles completamente descentralizados. Essa descentralização implica que os nós devem ser capazes de descobrir os demais no ambiente e manter conexão com os mesmos, para que possam então vir a fazer parte da rede. Dessa forma, um nó móvel poderia descobrir os dispositivos próximos para recuperar, por exemplo, os serviços e informações de contexto que mesmos disponibilizam. A descoberta de nós se torna bastante útil para nós recém chegados em um ambiente, seja o mesmo descentralizado ou não. Note que, em situações desse tipo, um nó não tem ciência de nenhum outro na rede. Portanto, o mesmo precisará primeiramente descobrir algum nó para que assim lhe seja permitido compartilhar e utilizar informações e serviços.

2.3 Considerações Sobre o Capítulo

Este capítulo apresentou características da Computação Ubíqua, bem como as redes de comunicação utilizadas na mesma. As redes, neste caso, são entendidas como um dos componentes centrais para a concretização dos estudos abordados nesse trabalho. No próximo capítulo serão apresentados os principais aspectos relacionados com a Computação Sensível ao Contexto, com intuito estabelecer uma fundamentação teórica, necessária para o entendimento da proposta desse trabalho.

3 SENSIBILIDADE AO CONTEXTO

A Computação Sensível ao Contexto investiga o emprego de informações que caracterizam a situação de uma interação usuário-computador no sentido de fornecer serviços adaptados a usuários e aplicações.

Este capítulo, estabelece as bases teóricas do trabalho, apresentando os conceitos fundamentais relacionados ao tema ”contexto”.

3.1 Definindo Contexto

A palavra ”contexto”no dicionário Houaiss significa a ”inter-relação de circunstâncias que acompanham um fato ou uma situação”. Por mais que essa definição forneça uma noção geral do significado de contexto, não mostra de que maneira esse conceito está relacionado com ambientes computacionais e sistemas de tecnologia da informação. A abrangência desse conceito leva a entender que, intuitivamente, contexto pode ser entendido como tudo que está ao redor de um sistema em questão, tudo que ocorre em um determinado ambiente.
Alguns pesquisadores, com o intuito de limitar a abrangência desse conceito, enumeraram exemplos de contextos. Schilit (SCHILIT, 1995) (SCHILIT B.N., 1994) divide contexto em três categorias:

- Contexto Computacional: conectividade de rede, custos de comunicação, largura de banda e recursos disponíveis como impressoras, processadores e memória;
- Contexto do Usuário: perfil do usuário, localização, pessoas próximas a ele, humor e outros;
- Contexto Físico: luminosidade, níveis de barulhos, condições do trânsito e temperatura.

Além disso, (CHEN G., 2000) defende a inclusão do Tempo (hora do dia, da semana, do mês e a estação do ano) como uma quarta categoria de contexto e introduz o conceito de Histórico de Contexto e a necessidade de armazenamento de informações contextuais como fonte de tomada de decisões e construção de aplicações sensíveis ao contexto. A definição mais referenciada na literatura de computação ubíqua para contexto é a defendida por (DEY, 2000):

”Contexto é qualquer informação que pode ser usada para caracterizar uma situação de uma entidade. Uma entidade é uma pessoa, um lugar, ou um objeto que é considerado

relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação."

(DEY, 2000) destaca que os contextos mais relevantes para um ambiente computacional são: a localização, a identidade, o tempo e a atividade de uma entidade, ou seja, a enumeração de exemplos de contexto ainda é bastante usada na literatura. Considerando a importância do ambiente ao seu redor e o quanto ele determina o comportamento de uma aplicação sensível ao contexto, (CHEN G., 2000) definem contexto da seguinte maneira:

"Contexto é o conjunto de estados e características de um ambiente que determina o comportamento de uma aplicação ou no qual um evento de uma aplicação ocorre e interessa ao usuário."

Podemos ainda entender contexto como "circunstâncias ou situações em que uma tarefa computacional está inserida", definição extraída de (HENRICKSEN K; INDULSKA, 2002).

Freqüentemente usadas como sinônimo de contexto, as informações contextuais são nada mais que as informações que caracterizam um determinado contexto. São elas as informações relevantes para se determinar o estado atual de um contexto em questão. Considere o contexto localização, as informações contextuais referentes a esse contexto são, por exemplo, a latitude e a longitude de uma entidade ou em que sala de um edifício se encontra uma determinada pessoa. Nesse trabalho, os termos contexto e informações contextuais são utilizados de forma semelhante, mas fica clara a diferenciação entre esses dois conceitos.

3.2 Características da Informações Contextuais

As informações contextuais possuem características bem peculiares que devem ser ressaltadas. De acordo com (HENRICKSEN K; INDULSKA, 2002), a natureza da informação contextual deve ser levada em conta ao se construir sistemas ubíquos e de computação sensível ao contexto. Eis alguns pontos que cabe destacar:

Características temporais da informação contextual: pode-se caracterizar uma informação contextual como sendo estática ou dinâmica. Informações contextuais estáticas descrevem aspectos invariáveis dos sistemas, como a data de aniversário de uma pessoa. Já as informações dinâmicas, as mais comuns em sistemas ubíquos, variam frequentemente. A persistência de uma informação contextual dinâmica é altamente variável, por exemplo, relações entre colegas e amigos podem durar por meses e anos, enquanto a localização e a atividade de uma pessoa frequentemente se alteram a cada minuto. A característica de persistência influencia e determina quando uma determinada informação deverá ser adquirida. Enquanto informações estáticas podem ser facilmente obtidas de maneira direta dos usuários das aplicações, mudanças freqüentes de contextos são detectadas através de meios indiretos como sensores.

Informação contextual é imperfeita: a informação pode estar incorreta se não refletir o verdadeiro estado do mundo que ela modela, inconsistente se contém informação contraditória, ou incompleta se sob alguns aspectos o contexto não é

reconhecido. Em ambiente extremamente dinâmico como o da computação ubíqua, a informação contextual rapidamente se torna obsoleta, não refletindo o ambiente que deveria representar. Isso ocorre pelo fato de frequentemente as fontes, os repositórios e os consumidores de contexto estarem distribuídos, gerando muitas vezes um atraso entre o envio e a entrega das informações contextuais. Além disso, os produtores de contextos, como sensores, algoritmos de derivação e usuários, podem prover informação imperfeita. Esse é particularmente um problema que ocorre quando uma informação contextual é inferida a partir de sensores de mais baixo nível; por exemplo, quando a atividade de uma pessoa é inferida indiretamente a partir de sua localização e do nível de ruído ao seu redor. Finalmente, quedas de canais de comunicação, interferências e outras falhas podem ocorrer no caminho entre e o envio e a entrega da informação contextual, perdendo parte do que foi enviado ou a informação por completo.

Contexto tem representações alternativas: a maioria das informações contextuais em sistemas sensíveis ao contexto é proveniente de sensores. Geralmente existe uma grande diferença entre aquilo que é lido nos sensores e a abstração entendida pelas aplicações. Essa diferença de abstração se deve aos tratamentos e processamentos que uma informação contextual deve passar. Por exemplo, um sensor de localização fornece as coordenadas geográficas de uma pessoa ou de um dispositivo, enquanto uma aplicação está interessada na identidade do prédio ou da sala em que o usuário está. Observe que os requisitos e níveis de abstração que uma informação contextual exige podem variar de uma aplicação para a outra. Portanto, um modelo de contexto deve suportar múltiplas representações do mesmo contexto em diferentes formas e em diferentes níveis de abstração, e ainda ser capaz de entender os relacionamentos entre essas representações alternativas. Informações contextuais são extremamente inter-relacionadas. Diversos relacionamentos entre as informações contextuais são evidentes, por exemplo, proximidade entre usuários e seus dispositivos. Entretanto, outros tipos de relacionamentos entre informações contextuais não são tão óbvios. As informações contextuais podem estar relacionadas entre si através de regras de derivação que descrevem como uma informação contextual é obtida a partir de uma ou mais informações.

3.3 Dimensões de Informação de Contexto

A partir das definições apresentadas nas seções anteriores, percebe-se que existe uma grande diversidade de informações que podem ser utilizadas como informações de contexto, diversidade essa que depende do domínio da aplicação em questão. Muitas aplicações sensíveis a contexto têm explorado informações de identidade e de localização de pessoas e objetos para proverem algum serviço útil a usuários, como as aplicações pioneiras Active Badge (SCHILIT, 1995) e ParcTab (SCHILIT B.N., 1994). Ambos protótipos utilizavam mecanismos emissores de sinais que forneciam a localização de pessoas em um edifício, além de identificarem essas pessoas em mapas eletrônicos periodicamente atualizados. Com tais informações era possível, por exemplo, realizar transferências automáticas de chamadas telefônicas.

Aplicações sensíveis a contexto mais recentes passaram a utilizar as facilidades do sistema de localização outdoor GPS (*Global Positioning System*), bastante utilizado no monitoramento de automóveis em cidades e rodovias. Por exemplo, o sistema CyberGuide (ABOWD G. D.; RODDEN, 2002) é utilizado como um guia turístico capaz

de escolher conteúdos áudio-visuais para serem exibidos conforme as informações de localização de pessoas. Com os avanços na área de comunicação por redes sem fio, novos sistemas sensíveis a contexto passaram a explorar informações de localização, como o sistema Guide, que utiliza sinais de redes 802.11 para identificar a localização de turistas ao longo de uma cidade e, a partir de sua localização, gerar roteiros personalizados.

No entanto, existem outras informações de contexto além de localização e identificação de pessoas e objetos. A maioria dos sistemas sensíveis a contexto não incorpora várias das informações disponíveis em um ambiente, como noções de tempo, histórico e dados de outros usuários. Em combinação com as características de aplicações sensíveis a contexto, (ABOWD G. D.; RODDEN, 2002) discutem a utilização de cinco dimensões semânticas de informações de contexto para auxiliar projetistas e desenvolvedores na especificação, na modelagem e na estruturação de informações de contexto de suas aplicações. Essas cinco dimensões semânticas são:

- *Who* (quem) - seres humanos realizam suas atividades e recordam de fatos passados com base na presença de pessoas e/ou objetos. Aplicações sensíveis a contexto devem, portanto, controlar a identificação de todas as entidades participantes de uma atividade no intuito de atender às necessidades de usuários. Informações de contexto de identificação podem incluir, entre outras, nome, email, senha, voz e impressão digital.

- *Where* (onde) - a mais explorada das dimensões de informações de contexto, a localização de entidades em ambientes físicos é normalmente associada a outras dimensões, como a dimensão temporal When (quando). Ao combinar essas duas dimensões, é possível explorar não apenas a mobilidade de usuários, mas também informações sobre sua orientação em um ambiente físico e, conse- qüentemente, fornecer serviços e/ou informações adaptados ao comportamento desses usuários. Informações de contexto de localização incluem, entre outras, latitude, longitude, altitude, cidade e posição relativa a objetos e pessoas.

- *When* (quando) - informações de contexto temporais podem ser usadas para situar eventos em uma linha do tempo, ou auxiliar na interpretação de atividades humanas e no estabelecimento de padrões de comportamento. Por exemplo, uma visita breve a uma página Web pode indicar falta de interesse do usuário com relação ao conteúdo da página. Já no caso de uma aplicação de monitoramento de pessoas idosas, essa aplicação verifica se os instantes ou intervalos de tempo das atividades do paciente são compatíveis com a rotina diária do mesmo. Nos casos em que há desvios de padrão, a aplicação deve notificar o médico de plantão. Informações de contexto temporais incluem, entre outras, data, hora, intervalos de tempo, dia da semana, mês e ano.

- *What* (o quê) - identificar o que um usuário está fazendo em um determi- nado momento pode ser uma tarefa complicada para uma aplicação em que atividades, não-previstas pelo projeto da aplicação, podem ser realizadas de forma concorrente. Configura-se, assim, como um dos principais desafios na computação sensível a contexto a obtenção de informações de contexto que possibilitem a interpretação correta da atividade de um usuário. Informações de contexto de atividades variam de aplicação para aplicação, por exemplo, escrever na lousa, anotar em um caderno, trabalhar em grupo e participar de uma reunião, palestra, ou operação cirúrgica.

- *Why* (por quê) - mais desafiador ainda que perceber e interpretar o que um usuário está fazendo, é entender o porquê de sua ação. Em geral, as informações de contexto de atividade (What) e de motivação (Why), por serem mais complexas, são obtidas por meio da combinação de informações de outras dimensões. O estado emocional de um usuário pode também ser indicativo de sua motivação para a realização de uma tarefa. Aplicações sensíveis a contexto podem obter, via sensores, informações que possam dar uma indicação do estado emocional de um usuário, por exemplo, o foco de atenção e a expressão facial, características de batimento cardíaco e níveis de pressão arterial, entonação vocal e ondas cerebrais do tipo alfa.

Essas cinco dimensões semânticas discutidas em (ABOWD G. D.; RODDEN, 2002) não sugerem completeza, mas sim, um conjunto básico de diretrizes a ser seguido no processo de construção de uma aplicação sensível a contexto. Nesse interim, (TRUONG K. N.; BROTHERTON, 2001) discutem uma dimensão semântica originada do domínio de aplicações de captura e acesso:

- *How* (como) - no contexto de aplicações de captura e acesso, esta dimensão fornece informações relativas a como recursos de um ambiente físico podem ser capturados e acessados. É importante que aplicações sensíveis a contexto te- nham informações não apenas do número e do papel dos dispositivos disponíveis para captura e acesso em um ambiente, mas também que estejam informados acerca das características funcionais de cada dispositivo para captura e acesso. Essas informações podem ser utilizadas, por exemplo, para a personalização de acesso a informações capturadas via dispositivos - por exemplo, os handhelds - com características de acesso bastante restritas, como tamanho de tela, quantidade de energia em bateria e suporte à entrada e saída de dados.

3.4 Modelagem de Contexto

Atualmente, o desenvolvimento de aplicações sensiveis ao contexto é uma tarefa complexa, o que torna o uso de técnicas de modelagem extremamente úteis. Contudo, os atuais modelos para desenvolvimento de software não oferecem suporte para o projeto e de tais aplicações e, sobretudo, tais técnicas de modelagem não provêem suporte para modelagem das informações contextuais (HENRICKSEN K; INDULSKA, 2002).

Existem inúmeras abordagens para modelar informações contextuais, dentre as quais pode-se ressaltar:

Modelos com métodos de marcação: seguem uma estrutura hierárquica de marcação com atributos e conteúdo. Em geral, utilizam-se de linguagens de marcação derivadas da Standard Generic Markup Language.

Modelos chave-valor: utilizam um modelo simples de atributo e valor, sendo fáceis de gerenciar, contudo têm pouco poder de expressão.

Modelos gráficos: baseados em notações gráficas, em geral são derivados de adaptações e extensões de modelos gráficos já difundidos, como UML, ORM ou o modelo Entidade Relacionamento.

Modelos orientados a objetos: esta abordagem tem a intenção de aplicar os principais benefícios do modelo orientado a objetos, notadamente, encapsulamento e reusabilidade, à modelagem de contexto. Nesses casos, o acesso às informações contextuais é feito somente através de interfaces bem definidas.

Modelos baseados em lógica: define-se o contexto de modo que se possa inferir expressões ou fatos a partir de um conjunto de outros fatos e expressões. Em geral, este modelo possui um alto grau de formalismo.

Modelos baseados em ontologias: uma ontologia é uma especificação de uma conceituação, isto é, uma descrição de conceitos e relações que existem entre eles, em um domínio de interesse. Nesse modelo o contexto é modelado em ontologias, construindo uma base de conhecimento do contexto.

Em (STRANG T; LINNHOFF-POPIEN, 2004) são avaliadas as abordagens citadas acima para modelagem de contexto, quando os seguintes critérios são considerados:

1. **Composição distribuída (cp):** a composição do modelo de contexto deve ser altamente dinâmica em termos de tempo, topologia de rede e origem, podendo estar distribuído em diversas localidades ao longo do tempo.

2. **Validação parcial (vp):** deve ser possível validar parcialmente o modelo, dado que nem todas as informações podem estar disponíveis ao mesmo tempo e que o conhecimento do contexto pode ser derivado da composição de outras informações distribuídas.

3. **Riqueza e qualidade da informação (rqi):** a qualidade e a riqueza das informações podem variar de acordo com o tempo e com o tipo de sensor. Daí que o modelo deve permitir anotações de qualidade e riqueza da informação de contexto representada.

4. **Incompletude e ambigüidade (ia):** as informações contextuais disponíveis em um dado momento podem ser incompletas ou ambíguas. Estas características devem ser cobertas pelo modelo.

5. **Nível de formalidade (nf):** a formalidade visa a dar um visão única do modelo; é altamente desejável que todos os participantes tenham a mesma interpretação. A formalidade permite, também, o processamento automático das informações de contexto diretamente do modelo, por exemplo, para validação.

6. **Aplicabilidade nos ambientes existentes (aae):** para uma boa aceitação, é importante que o modelo seja aplicável às infra-estruturas de suporte a contexto já existente.

Os resultados da análise realizada por (STRANG T; LINNHOFF-POPIEN, 2004) podem ser observados na Tabela 3.1. A notação apresentada atribui o sinal -"para o critério não satisfeito pelo modelo, e o sinal "+"para o critério atendido de maneira satisfatória. Sendo "++"para os critérios que são completamente satisfeito.

Tabela 3.1: Avaliação das abordagens para Modelagem de Contexto

Modelo	cp	vp	rqi	ia	nf	aae
Chave-Valor	-	-	-	-	-	+
Método de marcação	+	++	-	-	+	++
Gráficos	-	-	+	-	+	+
Orientação a objetos	++	+	+	+	+	+
Baseados em lógica	++	-	-	-	++	-
Baseados em Ontologia	++	++	+	+	++	+

3.5 Aquisição de Contexto

A aquisição de contexto está associada com a forma na qual as informações contextuais são obtidas, podendo ser sentida, derivada ou explicitamente provida (MOSTEFAOUI G. K., 2004).

Aquisição sentida: este tipo de informação pode ser adquirido do ambiente por meio de sensores(temperatura, nível de ruído, dispositivos presentes)

Aquisição derivada: este é o tipo de informação que pode ser obtida em tempo de execução. Por exemplo, é possível calcular a idade de uma pessoa baseada na sua data de nascimento.

Aquisição provida: informação que é explicitamente fornecida à aplicação. Por exemplo, os dados cadastrais de um usuário que é diretamente fornecido à aplicação por meio de um formulário.

Esta etapa de aquisição, entretanto, não é uma tarefa fácil, principalmente quando a informação é sentida. Isso ocorre devido à grande variedade de sensores. Além disso, informação contextual possui uma natureza dinâmica, sendo necessário que a aplicação gerencie todos esses aspectos.

3.6 Interpretação de Contexto

A interpretação de contexto pode ser entendida como o conjunto de métodos e processos que realizam a abstração, o mapeamento, a manipulação, a agregação, a derivação, a inferência e demais ações sobre as informações contextuais, com o propósito de facilitar o entendimento de um determinado contexto pelas aplicações e auxiliá-las na tomada de decisões. O processo de interpretação de contexto consiste na manipulação e refinamento das informações contextuais de um ambiente.

Em (DEY, 2000), a interpretação de contexto é vista como o processo de se elevar o nível de abstração das informações contextuais de um ambiente, ou seja, gerar uma informação contextual mais elaborada a partir de uma mais primitiva.

O processo de interpretação de contexto pode ser bastante simples como derivar o nome de uma rua a partir de suas coordenadas geográficas ou bastante complexo e oneroso como inferir o humor de um usuário baseado em seu perfil e na atividade em

que ele está realizando. Além disso, o ambiente em questão, o da computação ubíqua, é extremamente dinâmico e complexo. As informações contextuais podem estar espalhadas e distribuídas em qualquer lugar e com alto grau de mobilidade. Essa complexidade faz com que haja a necessidade de um suporte computacional às aplicações, de maneira a auxiliá-las na realização de interpretações de contextos. Tais atividades onerosas devem ser abstraídas das aplicações e o módulo Interpretador de Contexto torna-se, portanto, um componente essencial em uma plataforma de suporte a tais aplicações. Ele deve ser capaz de obter e prover informação contextual em diferentes níveis de abstração, conforme o desejo do usuário e de suas aplicações. Uma aplicação pode desejar tanto informações mais brutas, de mais baixo nível ou informações mais abstratas e elaboradas, de mais alto nível, provenientes de um processo de refinamento e interpretação.

3.7 Armazenamento de Informações Contextuais

A necessidade de manter o histórico de informações de contexto é um requisito ligado à aquisição de informações de contexto bem como à disponibilidade contínua dos componentes de captura de informações de contexto. Um histórico de contexto pode ser utilizado para estabelecer tendências e predizer valores futuros de informações contexto. Sem o armazenamento dessas informações, esse tipo de análise não é possível de ser realizado.

3.8 Arquitetura de Aplicações Sensíveis ao Contexto

Aplicações sensíveis ao contexto podem ser implementadas de diversas formas. A abordagem dependerá de requisitos e condições, tais como: a localização dos sensores, o número de usuários, a disponibilidade de recursos dos dispositivos utilizados, a facilidade para extensão do sistema. Com base nessas considerações três abordagens de arquiteturas para sistemas sensíveis ao contexto podem ser identificadas (CHEN, 2004):

- **Acesso Direto ao Sensor:** o software cliente obtêm a informação desejada diretamente do sensor, ou seja, não há qualquer camada adicional para obtenção e processamento dos dados do sensor. Esse aspecto dificulta a capacidade de expansão do sistema, por isso essa abordagem que tem sido pouco utilizada. Também, não é adequada para sistemas distribuídos, devido a natureza de seu acesso direto sem qualquer componente capaz de gerenciar múltiplos acessos concorrentes ao sensor.

- **Baseado em Middleware:** os modernos projetos de software usam métodos de encapsulação para separar a lógica de negócios da interface do usuário. A abordagem baseada em middleware introduz uma arquitetura em camadas para sistemas sensíveis ao contexto com a intenção de esconder os detalhes de baixo nível relativos à sensibilidade. Comparando com a abordagem de acesso direto ao sensor, esta técnica facilita a expansão do sistema, já que não há necessidade de modificações no código do cliente, bem como há uma simplificação na reutilização do código dependente de hardware, devido à rígida encapsulação.

- **Servidor de Contexto:** esta abordagem distribuída especializa a arquitetura baseada em middleware introduzindo um componente para o gerenciamento remoto de

acesso. Os dados obtidos dos sensores são movidos para um servidor de contexto com o objetivo de facilitar múltiplos acessos concorrentes. Além do reuso dos sensores, a utilização de um servidor de contexto tem a vantagem de retirar dos clientes operações que necessitam uso intensivo de recursos computacionais. Este é um aspecto importante, visto que a maioria dos dispositivos de borda usados em sistemas sensíveis ao contexto são dispositivos móveis com poder computacional limitado.

3.9 Uso de Aplicações Sensíveis ao Contexto

Uma das áreas de grande potencial de uso das aplicações sensíveis ao contexto que deve ser ressaltada é a área da Saúde. Considere, por exemplo, o cenário de um ambiente hospitalar. O trabalho clínico nos hospitais modernos é caracterizado por um alto grau de mobilidade (os médicos e enfermeiras, por exemplo, estão em constante trânsito pelos corredores e salas do hospital, e raramente ficam sentados num único lugar), freqüentes interrupções e muita colaboração ”ad hoc”entre colegas de diferentes especialidades. Essa natureza do trabalho clínico é pobremente suportada pelas arquiteturas cliente-servidor e pelos sistemas de informação hospitalar atuais, tipicamente desktop e voltados para o controle operacional e administrativo. Além disso, o contexto onde se dá a prática médica é simplesmente ignorado por essas arquiteturas e sistemas de informação. Localização imediata dos profissionais no seu ambiente de trabalho, disparo automático de ações em resposta a eventos de urgência, manipulação de informação contextual e uma engenharia de aplicações móveis baseada em atividades pode facilitar em muito o trabalho desses profissionais e melhorar a prática médica.

Alguns outros exemplos aplicações sensíveis ao contexto são listados a seguir:

- Aplicações turísticas. Mapas de atrações turísticas e posicionamento relativo dos turistas em relação às atrações que desejam visitar. As atrações que são visualizadas variam de acordo com o perfil de cada usuário;
- Aplicações policiais. Mapeamento e controle de viaturas e policiais em situações do dia-a-dia ou em situações mais perigosas como uma perseguição a um fugitivo;
- Aplicações voltadas ao meio ambiente. Controle e monitoramento de diversos animais ao mesmo tempo e em grandes áreas para fins de pesquisa e preservação do meio ambiente;
- Ambiente acadêmico e mapas de universidades. Mapas universitários e localização de alunos e professores em seu ambiente de trabalho;
- Monitoramento de pessoas que trabalham em uma empresa. Controle mais apurado dos trabalhadores em uma indústria ou em um escritório;
- Rastreamento de veículos em transporte de cargas e mercadorias. Monitoramento do posicionamento de veículos durante o percurso de um frete;

3.10 Considerações Sobre o Capítulo

Este capítulo resumiu o estudo feito sobre Computação Sensível ao Contexto, necessário para o desenvolvimento deste trabalho. O estudo contemplou aspectos como:

definições, características, dimensões, modelagem, aquisição, interpretação, armazenamento, arquitetura e cenários para aplicações sensíveis ao contexto. Estes aspectos serão utilizados para embazar a comparação feita no Capítulo 5, envolvendo Plataformas para o tratamento de Sensibilidade ao Contexto.

4 ONTOLOGIAS: CONCEITOS E TECNOLOGIAS

Ontologias têm sido largamente utilizadas em áreas como gerenciamento de conteúdo e conhecimento, comércio eletrônico e Web semântica. Particularmente, a comunidade científica tem apontado o uso de ontologias para lidar com alguns dos principais desafios relacionados à construção de ambientes ubíquos. De um modo geral, ontologias têm sido usadas para representar ambientes ubíquos, descrevendo, comumente, entidades envolvidas e suas respectivas propriedades. Elas definem principalmente os diferentes tipos de aplicações, serviços, dispositivos, usuários, entre outros. Além disso, estas ontologias definem descrições padrões para localização, atividades, informação sobre temperatura, etc. Neste capítulo, são exibidos os principais conceitos relacionados a este assunto, partindo do conceito de ontologia, passando pelos principais tipos de ontologias, benefícios advindos do uso de ontologias e finalmente descrevendo as principais linguagens para ontologias.

4.1 O Conceito de Ontologia

Embora a palavra ”ontologia” denote, em sua origem filosófica, uma teoria sobre a natureza do ser, para a Computação, ela vem sendo usada como um conjunto de entidades com suas relações, restrições, axiomas e vocabulário. Segundo (GRUBER, 1993), ”uma especificação de um vocabulário de representação para um domínio de discurso compartilhado - definições de classes, relações, funções e outros objetos - é uma ontologia”. O termo ontologia pode também ser definido a partir dos requisitos para possibilitar sua aplicação em informática. Sendo assim, uma ontologia pode ser definida como ”uma especificação explícita e formal de uma conceitualização compartilhada” (RUDI STUDER V. RICHARD BENJAMINS, 1998). Esclarecendo os requisitos desta definição, tem-se que (FREITAS, 2003):

- Por ”especificação explícita”, pode ser entendida como sendo definições de conceitos, instâncias, relações, restrições e axiomas.

- Por ”formal”, que é declarativamente definida através de uma linguagem formal, portanto, compreensível para agentes inteligentes e sistemas.

- Por ”conceitualização”, que se trata de um modelo abstrato de uma área de conhecimento ou de um universo limitado de discurso.

- Por ”compartilhada”, por tratar-se de um conhecimento consensual, seja uma terminologia comum da área modelada ou acordada entre os desenvolvedores dos agentes que se comunicam.

4.2 Tipos de Ontologias

Por se tratar de uma área da ciência que se aplica a qualquer parte do conhecimento, ontologias podem ser classificadas em uma escala de generalidade (MIZOGUCHI, 2004), de acordo com o propósito para o qual foi designada, como segue:

Ontologias de representação: definem as primitivas de representação - como frames, axiomas, atributos e outros - de forma declarativa. Esse tipo de ontologia serve para abstrair os formalismos de representação.

Ontologias gerais (ou de topo): trazem definições abstratas necessárias para a compreensão de aspectos do mundo (tempo, espaço, seres, coisas). Esses conceitos tipicamente são independentes de um problema particular ou domínio. Sendo assim, é bem razoável ter-se uma ontologia de alto-nível compartilhada por grandes comunidades de usuários.

Ontologias centrais (core ontologies) ou genéricas de domínio: definem os ramos de estudo de uma área e/ou conceitos mais genéricos e abstratos desta área. Por exemplo, a ontologia central de direito criada por (VALENTE; BREUKER, 1996), inclui conhecimentos normativos, de responsabilidade, reativos, de agências legais, comportamentos permitidos, etc. Esses conceitos e conhecimentos foram agrupados nesta ontologia para que ela sirva de base para a construção de ontologias de ramos mais específicos do direito, como direito tributário, de família e outros.

Ontologias de domínio: tratam de um domínio mais específico de uma área genérica de conhecimento, como direito tributário, microbiologia, etc. Ontologia de aplicação: procura solucionar um problema específico de um domínio, como identificar doenças do coração, a partir de uma ontologia de domínio de cardiologia. Normalmente, esse tipo de ontologia especializa conceitos tanto das ontologias de domínio, como também das de tarefas. Um exemplo disso é uma ontologia para uma aplicação que trabalhe com carros de luxo. Essa ontologia especializará conceitos da ontologia de veículos (que é uma ontologia de domínio).

Ontologias de tarefas: descrevem tarefas de um domínio (como processos, planos, metas, escalonamentos, etc.) com uma visão mais funcional, embora declarativa. Como pode ser percebido, no que foi descrito acima os tipos de ontologias estão listados em ordem decrescente de generalidade. É importante salientar também que nem todos os tipos são necessários para a construção de uma aplicação, sem mencionar a importância em manter as ontologias reusáveis, ou seja, fazer com que uma ontologia seja elaborada de forma que possa ser usada em diferentes situações.

4.3 Benefícios das Ontologias

Além dos benefícios advindos de uma abordagem declarativa, que descreve fatos e entidades acerca de um determinado domínio (metáfora do ”o que”), outros benefícios mais diretos, ligados à prática de construção de sistemas baseados em conhecimento, têm sido gerados. De início, o projeto Knowledge Sharing Effort (KSE) (ROBERT NECHES RICHARD FIKES, 1991) e suas ontologias contribuíram para uma maior cooperação entre os grupos de pesquisa responsáveis por manter as ontologias, da mesma forma como mantêm conhecimento, o que, tornando-se uma tendência, pode vir a provocar uma mudança cultural. Desde que foi criado o KSE, estão sendo definidas e mantidas ontologias extensíveis, abrangentes, gerais e muito detalhadas, por grupos de pesquisa, abarcando toda a pesquisa da área cujo conhecimento se deseja representar. Esta orientação ontológica trouxe muitos benefícios, alguns dos quais não previstos, e que só vieram frutificar na época de sua implementação. São eles:

- A oportunidade para os desenvolvedores de reusar ontologias e bases de conhecimento, mesmo com adaptações e extensões. O impacto sobre o desenvolvimento de sistemas baseados em conhecimento é substancial: a construção de bases de conhecimento redunda na tarefa mais cara e demorada de um projeto de sistemas especialistas e/ou agentes. As ontologias permitem ainda aos usuários efetuarem consultas, comparações, integração e verificação de consistência;

- A disponibilização de uma vasta gama de ”ontologias de prateleira”, prontas para uso, reúso e comunicação por pessoas e agentes. Hoje as ontologias mais maduras, algumas com mais de 2.000 definições, incluem metadados de imagens de satélites e para integração de bases de dados de genoma, catálogos de produtos, osciloscópios, robótica, semicondutores, terminologia médica, o padrão IEEE para interconexões entre ferramentas, entre outras;

- A possibilidade de tradução entre diversas linguagens e formalismos de representação de conhecimento. A tradução concretiza um ideal perseguido por gerações de pesquisadores de Inteligência Artificial. Ela facilita o reúso de conhecimento e pode vir a permitir comunicação entre agentes em formalismos diferentes, uma vez que este serviço encontra-se disponível para um número cada vez maior de formalismos de representação de conhecimento (para os formalismos tratados pela Ontolingua (A. FARQUHAR R. FIKES, 1996), ver Figura 4.1). Outra forma de alcançar esse intento são editores de ontologias em que pode-se escolher em que linguagem de representação será escrito o código gerado. No editor Protégé-2000 (NATALYA FRIDMAN NOY RAY W. FERGERSON, 2000), podem ser geradas ontologias em CLIPS, Jess, Prolog, XML, RDF, OIL, DAML-OIL e F-Logic;

- O acesso on-line a servidores de ontologias, capazes de armazenar milhares de classes e instâncias, que serviriam a várias empresas ou grupos de pesquisa, e que podem funcionar como ferramentas para manter a integridade do conhecimento compartilhado entre elas, garantindo um vocabulário uniforme; O mapeamento entre formalismos de representação de conhecimento, que, inspirado no componente de conectividade para sistemas gerenciadores de bancos de dados ODBC (Open Database Connectivity), integra dois formalismos criando uma interface interoperável

de acesso comum para eles, permitindo a um agente acessar o conhecimento de outro agente. O pacote gerado para implementar esta facilidade é chamado de OKBC (Open Knowledge Base Connectivity).

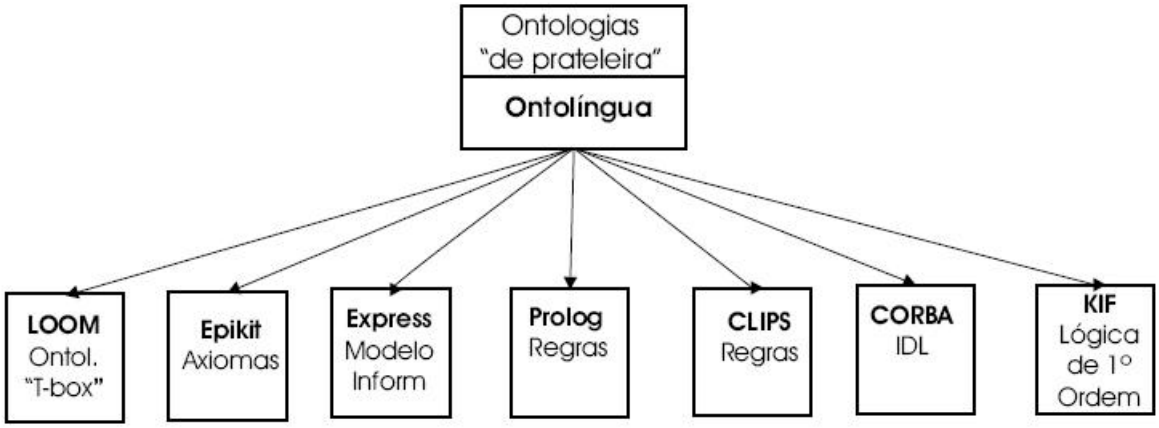


Figura 4.1: A Ontolíngua e os Formalismos para os quais podem ser Traduzidas as Ontologias (A. FARQUHAR R. FIKES, 1996)

4.4 Linguagens para Ontologias

Ontologias estão intimamente relacionadas com a linguagem usada para representá-las. Atualmente, existem algumas linguagens com esse propósito. A seguir, é apresentada uma visão geral sobre as principais linguagens, assim como das ontologias de cada linguagem.

4.4.1 RDF

RDF (Resource Description Framework) é uma linguagem de propósito geral para representar informação na Internet que baseia-se na idéia de identificar coisas através identificadores Web: os URIs (Uniform Resource Identifier).URIs são cadeias de caracteres utilizadas para identificar recursos na Web, como páginas, serviços, documentos, etc. Além dos identificadores Web (URIs), esta linguagem descreve recursos em termos de simples propriedades e valores. Isto permite que RDF represente recursos sob a forma de expressões sujeitopredicado- objeto:

1. O sujeito: é o recurso, ou seja, qualquer coisa que pode conter um URI, incluindo as páginas da Web, assim como elementos de um documento XML.

2. O predicado: é uma característica descritiva ou aspecto do recurso e por vezes expressa uma relação entre o sujeito e o objeto.

3. O objeto: é o objeto da relação ou o valor da característica descritiva RDF é um tipo de rede semântica (SOWA, 1992), sendo parecida, em termos de linguagem, com o Modelo Relacional. Isto implica que RDF é uma forma de representação de conhecimento que possui semântica auto-contida e oferece uma grande liberdade para criação de extensões personalizadas.

4.4.2 RDF Shema

RDF Schema (RDFs) é uma linguagem para representação de conhecimento que baseia-se na idéia de Frames (BUBLITZ, 2005). Ela tem sido usada para aumentar a expressividade de RDF, dispondo assim de um melhor suporte à definição e classificação. Este modelo organiza o conhecimento através de herança e de construtores de ontologias (frames, slots e facetas). Os frames são organizados em rede, significando que quando qualquer um deles for acessado, ligações com outros quaisquer, potencialmente importantes, estarão disponíveis, podendo ser visto como uma ”unidade de conhecimento”autosuficiente. Um frame é uma descrição de um objeto complexo. Ele é identificado por um nome e consiste de um conjunto de slots. Cada slot possui um nome único ao frame em que está definido, consistindo de um conjunto de facetas (atributos) de valores particulares. Sistemas baseados em frames permitem que os usuários representem o mundo com diferentes níveis de abstração, com ênfase sobre as entidades . Em adição ao que já é herdado pelo fato de basear-se em frames, RDFs dispõe de construtores de ontologias que tornam as relações menos dependentes de conceitos: usuários podem definir relações como uma instância de rdf:Property, descrever relações de herança como rdfs:subPropertyOf e então associar relações definidas com classes usando rdfs:domain ou rdfs:range (LI DING PRANAM KOLARI, 2005).

4.4.3 OWL

A OWL (Web Ontology Language) é uma linguagem para definir e instanciar ontologias na Web. Ela foi projetada para disponibilizar uma forma comum para o processamento de conteúdo semântico da informação na Web. Ela foi desenvolvida para aumentar a facilidade de expressar semântica disponível em XML, RDF e RDFs. Conseqüentemente, pode ser considerada uma evolução destas linguagens em termos de sua habilidade de representar conteúdo semântico da Web interpretável por máquinas. Já que a OWL é baseada em XML, a informação pode ser facilmente trocada entre diferentes tipos de computadores usando diferentes sistemas operacionais e linguagens de programação. Por ter sido projetada para ser lida por aplicações computacionais, algumas vezes considera-se que a linguagem não possa ser facilmente lida por humanos, porém esta é uma questão que pode ser resolvida utilizando-se de ferramentas adequadas. OWL vem sendo usada para criar padrões que forneçam um arcabouço para gerenciamento de ativos, integração empresarial e compartilhamento de dados na Web.

OWL atualmente tem três sub-linguagens (algumas vezes também chamadas de ”espécies”): OWL Lite, OWL DL e OWL Full. Estas três sublinguagens possuem nível crescente de expressividade, e foram projetadas para uso por comunidades específicas de programadores e usuários.

1. OWL Lite dá suporte aqueles usuários que necessitam principalmente de uma classificação hierárquica e restrições simples. Por exemplo, embora suporte restrições de cardinalidade, ela só permite valores de cardinalidade 0 ou 1. É mais simples fornecer ferramentas que suportem OWL Lite que seus parentes mais expressivos, e ela também permite um caminho de migração mais rápido de dicionários e outras taxonomias.

2. OWL DL suporta aqueles usuários que querem a máxima expressividade, enquanto mantém a computabilidade (garante-se que todas as conclusões sejam computáveis)

e decidibilidade (todas as computações terminarão em tempo finito). OWL DL inclui todas as construções da linguagem OWL, porém elas somente podem ser usadas com algumas restrições (por exemplo, embora uma classe possa ser subclasse de muitas classes, uma classe não pode ser instância de outra classe). OWL DL é assim chamada devido a sua correspondência com as lógicas de descrição, um campo de pesquisa que estudou a lógica que forma a base formal da OWL.

3. OWL Full é direcionada àqueles usuários que querem a máxima expressividade e a liberdade sintática do RDF sem nenhuma garantia computacional. Por exemplo, em OWL Full uma classe pode ser tratada simultaneamente como uma coleção de indivíduos e como um único indivíduo. OWL Full permite que uma ontologia aumente o vocabulário pré-definido de RDF ou OWL.

4.5 Considerações Sobre o Capítulo

Nesse capítulo foram apresentados os conceitos e as tecnologias relacionadas com ontologias. Também foram abordadas as principais motivações para o uso de ontologias. Quanto as linguagens utilizadas para construção de ontologias, destacam-se as destinadas a aplicaçoes para Web Semântica, especialmente a OWL. No capítulo a seguir, serão apresentadas Plataformas Sensíveis ao Contexto e uma tabela comparativa entre as mesmas, o estudo sobre Especificação de informações de contexto por ontologias, será um dos aspectos analizados.

5 PLATAFORMAS SENSÍVEIS AO CONTEXTO

Este capítulo apresenta algumas iniciativas de projetos de plataformas de serviços context-aware referenciados na literatura da área, que são exemplos representativos de sistemas de middleware de apoio ao desenvolvimento e execução de aplicações sensíveis ao contexto. As arquiteturas apresentadas a seguir formam um conjunto representativo dos trabalhos que vêm sendo desenvolvidos nos últimos anos na direção de infra-estruturas de suporte à computação context-aware.

5.1 Context Kernel

O Context Kernel foi desenvolvido originalmente por (ARRUDA JR., 2003) em seu trabalho de mestrado, com o objetivo de dar suporte ao desenvolvimento de aplicações sensíveis ao contexto. O Context Kernwel é um web service que fornece o armazenamento, recuperação e intercâmbio de informações de contexto entre aplicações por meio de operações XML que implementam as dimensões semânticas discutidas em (ABOWD G. D.; RODDEN, 2002) e (TRUONG K. N.; BROTHERTON, 2001): tempo, localização, identificação, modo de captura e acesso de dados, atividade e motivação.

O Context Kernel classifica as dimensões de tempo, localização, identidade e modo de captura e acesso de dados como informações primitivas (ou atômicas), enquanto que atividade e motivação como informações derivadas por serem obtidas a partir da combinação de outras dimensões (primitivas e/ou derivadas) na forma de regras de inferência (ARRUDA JR., 2003).

Vale destacar que a abordagem de web services empregada no Context Kernel traz como vantagens:

- utilizar a infra-estrutura da Internet e especificações XML que permitem comunicação transparente em ambientes heterogêneos de hardware e software;
- assumir a responsabilidade pelo processamento de informações de contexto.

Considerando isso, o Context Kernel permite que as aplicações armazenarem, recuperarem e compartilharem informações de contexto utilizando a Web como plataforma de intercâmbio. Deste modo, o Context Kernel contribui para simplificação do processo de desenvolvimento de infra-estruturas computacionais para suporte ao desenvolvimento de aplicações sensíveis ao contexto.

5.1.1 Arquitetura

O Context Kernel permite que aplicações não somente armazenem e recuperem, mas também compartilhem informações de contexto por meio da Web. Assim, beneficia-se da especificação SOAP para estruturar o formato das mensagens, da linguagem WSDL para definir as mensagens e da linguagem XML Schema para evitar ambigüidade na representação dos tipos de dados em diferentes plataformas que processarão as mensagens.

A Figura 5.1 ilustra o fluxo de informação por meio dos passos numerados de 1 a 5. No passo 1, o Context Kernel publica a descrição de seus serviços em um documento público WSDL. No passo 2, o desenvolvedor de aplicações recupera o conteúdo do documento WSDL. Analisando esse documento, o desenvolvedor obtém a descrição dos serviços disponibilizados pelo Context Kernel, suas localizações na Web e o formato das mensagens para invocá-los. Com base nessas informações, a aplicação é construída de modo a enviar requisições HTTP utilizando o protocolo SOAP para o empacotamento das mensagens. No passo 3, a aplicação requisita os serviços disponibilizados pelo Context Kernel. No passo 4, o Context Kernel comunica-se com sua base de dados e armazena/recupera as informações que são enviadas/requisitadas pelas aplicações. Dependendo do resultado da análise dessas informações e do sucesso no armazenamento ou recuperação da base de dados, o Context Kernel envia uma mensagem de retorno apropriada no passo 5.

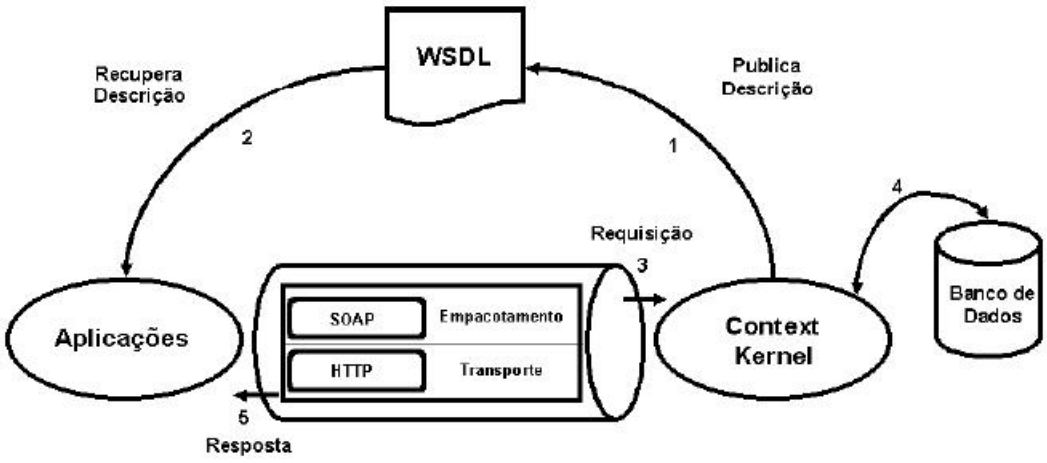


Figura 5.1: Fluxo de Informação do Context Kernel (ARRUDA JR., 2003)

Atualmente as operações disponibilizadas pelo Context Kernel são as de registro, armazenamento, consulta, recuperação e notificação. A seguir estão descritos cada uma dessas operações (ARRUDA JR., 2003):

Registro de aplicações: O registro de informações, disponibilizado pela operação InfoApp, permite que aplicações se registrem como usuárias do Context Kernel. Uma aplicação deve informar seu nome, uma descrição, seu endereço na Web, o número e data da sua última versão, o domínio (comercial, acadêmico, demonstração, gratuito ou versão para avaliação) e o nome e o endereço eletrônico dos seus desenvolvedores.

Armazenamento de informações de contexto: A operação PutData é a responsável pelo armazenamento das informações primitivas, enquanto a operação PutRules é a responsável pelo armazenamento das regras de contexto. A operação PutData permite que aplicações enviem várias sentenças sobre as dimensões contextuais, enquanto que a operação PutRules, requer que as aplicações enviem um ou mais dados primitivos que fazem parte da premissa, ou um ou mais dados derivados que fazem parte da inferência da regra.

Recuperação de informação de contexto: O Context Kernel disponibiliza três operações para a recuperação de informações de contexto: StatusApp, StatusApps e GetRules. As operações StatusApp e StatusApps permitem a recuperação das informações de registro de uma dada aplicação e de todas as aplicações registradas no Context Kernel, respectivamente. Por motivos de segurança o identificador privado não é recuperado, apenas o identificador público. A operação GetRules é a responsável pela recuperação das regras de contexto armazenadas na base de dados, sendo necessário, para isso, informar o identificador público da aplicação.

Consulta de informações de contexto: As operações GetAny e GetInverse são as responsáveis pelas operações de consultas. A operação GetAny permite às aplicações realizarem consultas com base nos valores das premissas associando dimensões primitivas e derivadas. Ainda, pode-se requisitar o número máximo de respostas retornadas, quais as dimensões de resposta, especificar o tipo da dimensão e usar os operadores booleanos AND e OR. A operação GetInverse permite às aplicações recuperarem informações de contexto utilizando uma inferência para obter as correspondentes premissas.

Notificação de eventos e validação de regras: O Context Daemon é uma operação de notificação de eventos que estende o modelo de gerenciamento de informações de contexto proposto pelo Context Kernel (JARDIM, 2003). Esse modelo baseia-se nos conceitos de regras de contexto e de informações primitivas. Uma notificação de evento é decorrente da validação de uma regra. Definimos validação de uma regra como o processo resultante da igualdade léxica decorrente da comparação das premissas de uma informação primitiva com as premissas de uma regra de contexto. A notificação consiste em tornar uma aplicação ciente que uma regra registrada por ela foi validada. Uma aplicação poderia armazenar a regra. No Context Kernel a regra é armazenada na base de dados, mas como ela contém o atributo ”notify”com valor ”true”, o que indica que a aplicação tem interesse em receber notificação de eventos para essa regra, essas informações contextuais são enviadas ao Context Daemin. As informações recebidas pelo serviço PutData, não importando qual a aplicação as originou, também são enviadas para o Context Daemin para tentativa de validação das regras. Se a informação primitiva fosse recebida pelo Context Kernel, ela validaria a regra, resultando em uma notificação do evento ocorrido, realizada pelo Context Daemin, para a aplicação que registrou a regra. A mensagem de notificação enviada pelo Context Daemin contém as premissas e inferências da regra, porque não é o objetivo aqui deixar sobre responsabilidade do Context Kernel as inferências sobre as regras de contexto.

5.2 Context Toolkit

O Context Toolkit é um framework conceitual de suporte à construção, ao desenvolvimento, à organização e ao funcionamento de aplicações sensíveis ao contexto. O toolkit foi desenvolvido de maneira a atender alguns requisitos de tratamento de contextos e informações contextuais (DEY, 2000):

- Separação de preocupações: separar a aquisição da manipulação e do tratamento de contextos;
- Interpretação de contexto: estender mecanismos de busca, notificação, interpretação e inferência de contextos para permitir o uso uniforme e adequado da informações provenientes de um ambiente dinâmico e altamente distribuído;
- Comunicação transparente e distribuída: os responsáveis pelo desenvolvimento das aplicações e os designers de sensores de informações não devem se preocupar com detalhes de protocolos de comunicação e com os níveis de codificação e implementação da transmissão de informação contextual;
- Disponibilidade permanente na tarefa de aquisição de contexto: as aplicações sensíveis ao contexto não devem se relacionar apenas com um único elemento específico que provê um determinado dado contextual, mas devem ser capazes de acessar aqueles elementos que estão disponíveis no momento. Além disso, múltiplas aplicações podem acessar a mesma informação ao mesmo tempo. Assim, os componentes que realizam a aquisição de contextos devem ser executados independentemente das aplicações clientes que os utilizam;
- Histórico e armazenagem de contexto: necessidade constante de se manter e guardar um histórico das informações contextuais lidas e processadas pelas aplicações;
- Descoberta de recursos: descoberta de novos e eficientes provedores de contexto (ou pelo menos suas interfaces de software). As aplicações precisam saber que tipo de informação contextual um provedor de contexto oferece, onde ele se localiza e qual a forma de comunicação de dados que ele utiliza (tipo de protocolo, linguagem ou mecanismos de comunicação).

5.2.1 Arquitetura

Os elementos de apoio à construção das aplicações são divididos em cinco categorias de componentes dentro do framework conceitual. São eles:

Context Widgets: são componentes de software que provém às aplicações acesso às informações contextuais em seus ambientes de operação. Eles abstraem a complexidade de uso e comunicação de sensores e fornecem informações em formatos adequados para o entendimento dos dados pelas aplicações. Os Widgets são blocos de componentes que podem ser organizados e reutilizados de acordo com a necessidade das aplicações. Do ponto de vista da aplicação, os Widgets fornecem um mecanismo de acesso uniforme e encapsulado das informações contextuais.

Interpreters: são responsáveis por elevar o nível de abstração de um dado contexto. Por exemplo, uma informação de localização pode ser expressa tanto na forma de latitude e longitude quanto em níveis de abstração mais elevados como uma cidade ou uma rua. Interpreters podem ser compostos por múltiplas camadas de abstração, de acordo com a necessidade das aplicações. Eles também são capazes de inferir novas informações contextuais a partir de várias fontes de contexto.

Aggregators: coletam informações de várias fontes que são logicamente relacionadas a um tipo de contexto em um repositório e as torna disponível em um componente único de software.

Services: são componentes que são capazes de executar determinadas ações de acordo com o interesse e a necessidade de uma dada aplicação. Quando uma determinada combinação de condições acontece, uma ação pré-definida deve ser tomada a pedido da aplicação.

Discoverers: são responsáveis pela manutenção de um registro as informações de todos os componentes e o que cada um pode realizar. Quando um Widget, um Interpreter, um Aggregator ou um Service é implementado ou instanciado no framework, um Discovery é notificado dessa presença, registrando sua existência e a melhor maneira de acessá-lo.

Pode-se dizer que o Context Toolkit é formado pelos Context Widgets e por uma infraestrutura distribuída dos demais elementos que recebem e manipulam as informações lidas pelos Widgets. A Figura 5.2, extraída de (DEY, 2000), mostra o exemplo de uma configuração dos componentes apresentados.

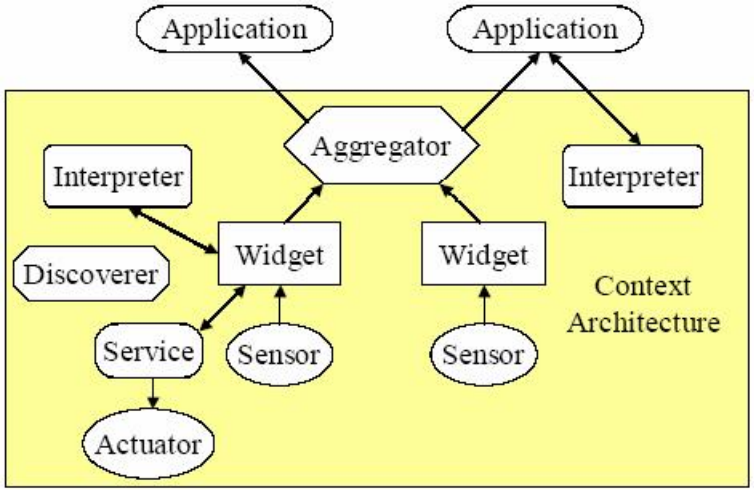


Figura 5.2: Componentes do Context Toolkit (DEY, 2000)

Os componentes são: dois sensores, dois Widgets, um Aggregator, dois Interpreters, um Service, um Discovery e duas aplicações quaisquer. Quando um dos componentes citados se torna disponível para utilização, ele se registra ao Discovery. Isso permite que Aggregators encontrem Widgets e Interpreters para se comunicarem ou permite às aplicações encontrarem Aggregators, Widgets e Interpreters que sejam relevantes. Os

sensores fornecem dados aos Context Widgets, que armazenam essas informações ou as envia a um Interpreter para gerar abstrações de alto nível do que foi recebido e, posteriormente, tornar essas informações mais elaboradas disponíveis aos demais componentes do toolkit e às aplicações. Um Aggregator coleta e armazena informações contextuais lidas dos Widgets. Finalmente, as aplicações podem requisitar aos Aggregators, aos Widgets ou aos Interpreters as informações contextuais que desejarem. Todos os componentes executam de maneira independente das aplicações, permitindo o fornecimento ininterrupto de contexto e informação contextual a diversas aplicações ao mesmo tempo.

5.3 Infraware

A plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006) é um *middleware* baseado em Web Services com suporte arquitetural para o desenvolvimento, construção e execução de aplicações móveis sensíveis ao contexto. A arquitetura conceitual da Infraware estende, em vários aspectos, a da plataforma WASP (WASP, 2003), um projeto holandês desenvolvido pela University of Twente, Telematica Instituut e Ericsson. A plataforma WASP concentra-se na interface aplicação-plataforma, definindo uma linguagem para especificar como ela deve reagir a uma correlação de eventos. A Infraware, por sua vez, foi definida visando o atendimento a vários requisitos funcionais presentes em ambientes sensíveis ao contexto e a integração desses em uma infra-estrutura única, formando uma arquitetura flexível e adequada ao desenvolvimento de aplicações ubíquas reais, em domínios variados. Por exemplo, a Infraware está sendo usada como base para o desenvolvimento de aplicações móveis, especificamente, na área da Saúde.

Uma característica marcante da Infraware é o uso de conceitos da Web Semântica: ontologias especificam modelos formais extensíveis que descrevem não somente o domínio das aplicações, mas também os serviços. Essa abordagem diferenciada provê meios de configurar as interações aplicação-plataforma em run-time. A plataforma também pode ser customizada pela adição de novos serviços e entidades estendendo-se as ontologias. Adicionalmente, a adoção de Web Services como tecnologia de distribuição permite que aplicações acessem os serviços oferecidos através de protocolos da Internet e facilita a inclusão de novos serviços à plataforma por terceiros. Essa flexibilidade torna a Infraware adequada ao desenvolvimento de uma larga gama de aplicações em cenários reais.

5.3.1 Arquitetura

A Infraware apresenta uma camada específica para o recebimento e o tratamento das subscrições das aplicações à plataforma e trata do controle de acesso e privacidade de maneira especializada através de um módulo direcionado a tal propósito. A plataforma também resolve o problema do acesso e integração de dados heterogêneos através de uma infra-estrutura dedicada, e é capaz de manipular, derivar e interpretar semanticamente informações de contexto de domínios variados. Além disso, aborda o problema da resolução de conflitos entre aplicações de maneira diferenciada através de um componente coordenador. A Figura 5.3 ilustra a arquitetura geral da plataforma.

Os seus principais componentes são descritos a seguir.

Gerente de Subscrição: Este componente fornece uma interface que permite aos

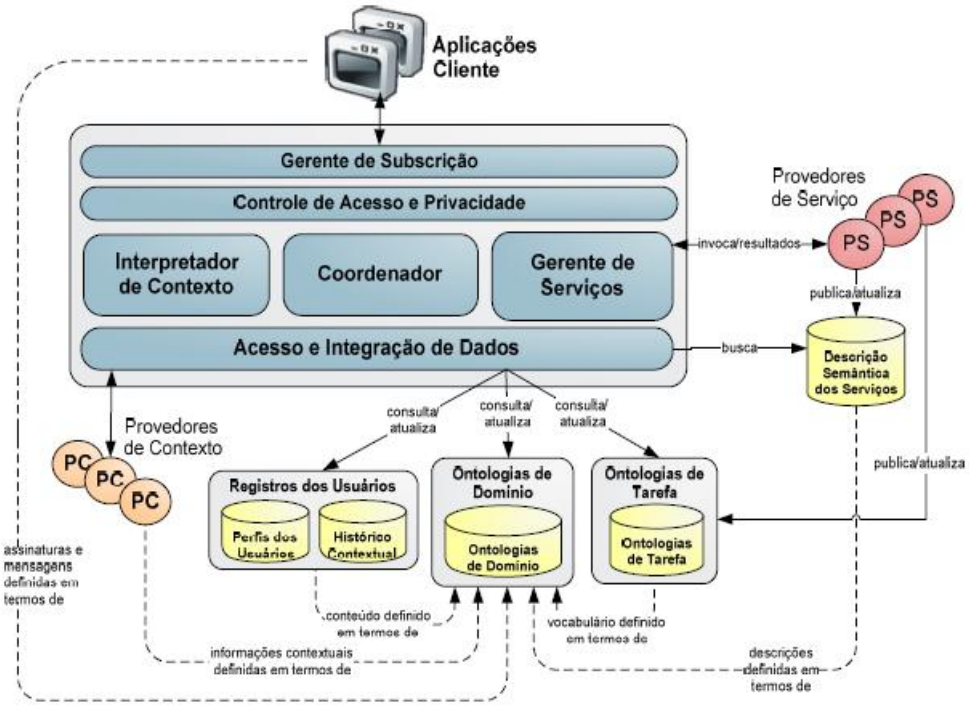


Figura 5.3: Plataforma Infraware (PEREIRA FILHO J. G.; PESSOA, 2006)

clientes do Infraware removerem, adicionarem ou atualizarem pedidos de subscrições. Ele se vale de sentenças e expressões trocadas entre as aplicações e a plataforma, através de uma linguagem específica voltada a tal propósito.

Controle de Acesso e Privacidade: O módulo de Controle de Acesso e Privacidade atua como um filtro sobre o fluxo de dados entre a plataforma e as aplicações, com base em um conjunto de restrições envolvendo preferências e políticas de privacidade dos usuários e das aplicações. O uso de políticas de privacidade também permite estabelecer limites de visibilidade para os dados coletados.

Interpretador de Contexto: Ele é responsável pela manipulação, derivação, refinamento e inferência de contextos mais elaborados a partir de informações contextuais primitivas, com a finalidade de tornar as informações disponíveis para as aplicações de forma transparente, reduzindo a complexidade no tratamento dos dados, (CALVI C. Z., 2005). Na versão atual, esse componente utiliza grafos acíclicos dirigidos como estrutura básica para manipulação e interpretação das informações contextuais. Além disso, utilizam-se ontologias para manipulação e inferência de novas informações, garantindo uma representação compartilhada e reutilizável dos dados.

Acesso e Integração de Dados: O CoDIMS (Configurable Data Integration Middleware System) é um middleware para integração de dados baseado em frameworks e componentes. O módulo de Acesso e Integração de Dados estende o CoDIMS, provendo funcionalidades capazes de tratar e manipular informações oriundas de diversas fontes de contexto, oferecendo também uma interface homogênea e transparente de acesso aos dados.

Gerente de Serviços: É responsável pelas atividades de publicação, descoberta, seleção e composição de serviços. A abordagem utilizada baseia-se na descrição e seleção semântica de serviços, na descoberta dinâmica e na utilização de mecanismos de segurança. A seleção baseada em descrição semântica possui a vantagem de permitir a descoberta de serviços que atendam às características descritas pelo usuário, em vez de limitar a seleção a parâmetros pré-estabelecidos e restritos ao protocolo.

Coordenador: Exerce atividades de gerenciamento, geração e disparo de planos de ações executados pelos outros componentes do middleware. Suas principais funções estão relacionadas com a monitoração e o controle do estado geral da plataforma.

5.4 Moca

A MoCA, Mobile Collaboration Architecture (SACRAMENTO et al., 2004), é uma arquitetura de middleware para o desenvolvimento de aplicações colaborativas e cientes de contexto para computação móvel. É constituída por APIs para implementação de clientes e servidores, serviços básicos para aplicações colaborativas e um framework para implementação de proxies.

A MoCA define que cada aplicação tem três partes: um servidor, um proxy e um ou vários clientes. O servidor e o proxy executam em nós fixos, enquanto a parte cliente executa em nós móveis. O proxy realiza a intermediação de toda a comunicação entre o servidor e o cliente. É nele que está a lógica de adaptação para a aplicação clienteservidor a qual está vinculado.

Para a construção desses proxies, a MoCA disponibiliza um framework denominado ProxyFramework. Este framework provê mecanismos de acesso às informações de contexto relacionadas à interação cliente-servidor, e também define a programação de adaptações disparadas pelas mudanças de contexto. Ele implementa algumas das características mais comuns, quando se usa uma abordagem baseada em proxy, como meios para lidar com a mobilidade dos clientes e conectividade intermitente.

5.4.1 Arquitetura

Os serviços que formam a arquitetura MoCA destinam-se a dar suporte ao desenvolvimento e a execução de aplicações colaborativas cientes de contexto. São eles:

Monitor: é um processo que executa em segundo plano em cada dispositivo móvel. Ele coleta informações sobre o ambiente e o estado de execução no dispositivo e os envia ao CIS (Context Information Service).

Configuration Sevice (CS): é responsável por armazenar e gerenciar as informações de configuração para todos os dispositivos móveis. Ele faz uso de tabelas hash, indexadas pelo endereço MAC, para armazenar o endereço do CIS e do DS (Discovery Services), aos quais um dado dispositivo está associado.

Discovery Services (DS): armazena informações das aplicações e serviços registrados na MoCA.

Context Information Services (CIS): recebe e processa as informações de estado enviadas por cada Monitor. Faz uso do modelo Publisher/ Subscribe para receber inscrições de proxies e enviar eventos a cada um desses proxies, com informações sobre mudanças no estado dos dispositivos de interesse.

Location Inference Service (LIS): Este serviço fornece a localização aproximada dos dispositivos, tomando como base o padrão de sinais recebidos pelo dispositivo de diversos pontos de acesso IEEE 802.11.

Na Figura 5.4 é ilustrada a arquitetura MoCA e a interação entre seus diversos componentes durante o registro e execução de uma aplicação.

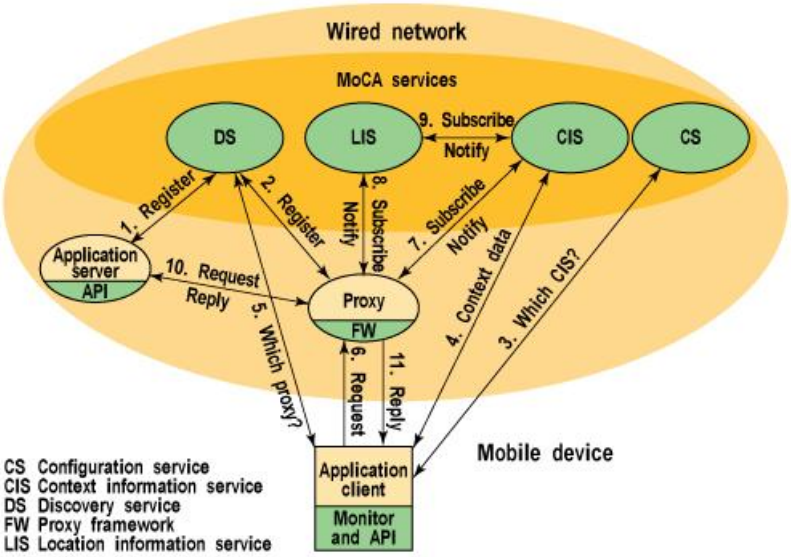


Figura 5.4: Serviços da Arquitetura Moca (SACRAMENTO et al., 2004)

Primeiramente, a parte servidora da aplicação se registra no DS (1), informando os nomes e as propriedades dos serviços que implementa. Cada um dos proxies para esta aplicação realiza um processo semelhante (2). O Monitor, que executa em cada nó móvel, quando iniciado, obtém do CS o endereço do CIS (3), para o qual, periodicamente, enviará as informações de contexto (4).

A partir desse momento, a aplicação cliente entra em contato com o DS, com a finalidade de descobrir como acessar um serviço colaborativo na rede (5). O DS retorna os dados do proxy para a aplicação cliente. Este será o proxy que fará a ponte entre a aplicação cliente e a parte servidor. A seguir, todas as requisições são encaminhadas ao proxy (6), o qual as processa e as adapta, caso necessário, e as encaminha ao servidor (10). Nesse ponto o proxy se inscreve junto ao CIS para receber notificações de mudança no estado do cliente (7).

Para aplicações que requerem informações de localização, oproxy irá se inscrever também no LIS (8), o qual se inscreve no CIS (9) para receber informações sobre a potência dos sinais dos pontos de acesso no cliente em questão.

Quando a parte servidora recebe a requisição, ela é processada e o resultado enviado ao proxy (10), o qual, de modo semelhante ao envio, processa e adapta os resultados

Tabela 5.1: Comparação entre Plataformas Sensíveis ao Contexto

Plataforma	TC1	TC2	TC3	TC4	TC5	TC6	TC7
Context Kernel	OWL-Ontologias	tradutor de contexto	serviço de inferência de contexto	N	N	*arq DB*	*N*
Context Toolkit	XML	widgets	interpretador	XML sobre HTTP	S	*DB*	*S*
Infraware	OWL-Ontologias	I	Motor de inferência	S	S	*DB*	*N*
Moca	OWL-Ontologias	Servidor-proxy	interpretador	ProxyFramework	S	*S*	*S*

de acordo com o estado atual do cliente, obtido do CIS. Finalmente, a resposta é enviada ao dispositivo móvel (11).

5.5 Tabela Comparativa

O objetivo principal desse trabalho é analizar as plataformas sensíveis ao contexto. Nesta seção é apresenta uma análise comparativa entre as quatro plataformas detalhadas anteriormente. A tabela 5.1 ilustra a comparação entre as plataformas, onde:

- TC1 - Especificação de informações de contexto
- TC2 - Separar aquisição de utilização de informações de contexto
- TC3 - Interpretação de informação de contexto
- TC4 - Comunicação distribuída e transparente
- TC5 - Disponibilidade contínua de componentes de captura de informação de contexto
- TC6 - Armazenamento de informações de contexto
- TC7 - Descoberta de recursos

Vale ressaltar que aqueles requisitos não contemplados por uma determinada plataforma apresentam o valor N associado.

Os requisitos TC5 e TC7 descrevem apenas se são considerados (S) sim ou (N) não por cada um das plataformas. Quando não é possível determinar se um requisito é tratado por uma plataforma, é assumido que a informação está (I) indisponível.

5.6 Considerações Sobre o Capítulo

As plataformas apresentadas neste capítulo tiram o proveito do uso de contexto para prover suporte para execução de aplicações Sensíveis ao Contexto. Dentro

os benifícios do estudo dessas plataformas foi a tabela comparativa apresentada nesse capítulo, onde foram analizados pontos em comum entre as mesmas, que são de grande contribuição para continuidade dos esforços de pesquisa.

6 CONSIDERAÇÕES FINAIS

Este trabalho apresentou uma revisão dos fundamentos teóricos sobre as áreas de Computação Ubíqua, Computação Sensível ao Contexto e Ontologias.

Essas áreas são alvos de intenso esforço em pesquisas. Em Computação Sensível ao Contexto, várias pesquisas têm resultado em propostas de infra-estruturas para suportar o desenvolvimento de aplicações que antecipem as necessidades dos usuários e reajam automaticamente de forma pouco intrusiva diante de uma situação.

Desde que Mark Weiser concebeu sua visão de ubiqüidade, importantes evoluções no *hardware* tem sido obtidas, permitindo a criação de dispositivos menores e mais portáteis, sensores e dispositivos de controle com crescente poder de processamento e a padronização das tecnologias para comunicação sem fio. Com isso, estão sendo criadas as condições para permitir a premissa básica da computação ubíqua, ou seja, o acesso do usuário ao seu ambiente computacional a qualquer hora, em qualquer lugar, independente de dispositivo.

Na Computação Ubíqua um aspecto fundamental relaciona-se ao monitoramento e a manipulação das informações contextuais. Neste sentido, a Computação Sensível ao Contexto é um paradigma computacional que se propõe a permitir que as aplicações tenham acesso e tirem proveito de informacões que digam respeito às computações que realizam, buscando otimizar seu processamento.

Um sistema é sensível ao contexto, se ele usa contexto para prover informações ou serviços ao usuário. Suas aplicações são capazes de modificar seu comportamento baseado nas informações de contexto ou são aplicações que mostram ao usuário informações de contexto.

Outro aspecto abordado nesse trabalho foi o uso de Ontologias para lidar com os principais desafios relacionados à construção de ambientes ubíquos sensíveis ao contexto, descrevendo, comumente, entidades envolvidas e suas respectivas propriedades.

As Ontologias definem principalmente os diferentes tipos de aplicações, serviços, dispositivos, usuários, entre outros. Além disso o uso de Ontologias servem para modelar o reconhecimento e processamento das informações contextuais. As Ontologias como técnica de modelagem é justificado pelas suas características de formalidade, semântica explícita e abastração de implementação, que são necessárias para a modelagem das informações contextuais.

Entende-se como contribuição principal deste Trabalho Individual, o relacionamento e comparações de plataformas que suportam a construção de aplicações sensíveis ao contexto. Através de uma tabela comparativa foi demonstrado as interligações de sensibilidade ao contexto nas plataformas estudadas.

Este estudo vem sendo oportuno para o aprofundamento nas áreas de Computação Ubíqua, Sensibilidade ao Contexto e o uso das Ontologias e suas tecnologias. Foi possível constatar que o uso de Ontologias é utilizado na grande maioria das plataformas para especificação das informações de contexto. Nas plataformas comparadas todas possuem algum modo de armazenamento das informações contextuais e a captura dessas informações estão presentes na maioria das plataformas comparadas. Estes foram alguns dos resultados constatados que tornaram oportuno os esforços dessa pesquisa.

Como trabalhos futuros, destaca-se a continuidade da pesquisa do Mestrado em Ciência da Computação, que ocorre dentro do mesmo escopo geral deste trabalho individual, ou seja, a Sensibilidade ao Contexto na Computação Ubíqua. Assim, pretende-se integrar ao *middleware* EXEHDA, tecnologias de Web Semântica no desenvolvimento de aplicações sensíveis ao contexto através da modelagem e implementação de ontologias como extensão a uma infra-estrutura de gerencimento de informações contextuais.

REFERÊNCIAS

A. FARQUHAR R. FIKES, J. R. **The Ontolingua Server**: a Tool for Collaborative Ontology Construction. Universidade de Stanford, EUA: [s.n.], 1996.

ABOWD G. D., M. E. D.; RODDEN, T. **IEEE Pervasive Computing**. [S.l.]: The human experience, 2002.

AL., C. N. et. **Handling exceptional conditions in mobile collaborative applications**: As exploratory case study. [S.l.]: In: 15th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Entreprises, 2006. 347-142p.

ARRUDA JR., C. **Context Kernel**: um Web Service baseado nas dimensões de informação de contexto. [S.l.: s.n.], 2003. 85pp.

BROENS, T. P. S. S. M. K. J. D. P. **Context-Aware, Ontology-Based Service Discovery**. [S.l.]: In. Second European Symposium, 2004. 72-83p.

BROLL, G. S. S. R. E. P. M. H. J. W. M. S. A. **Supporting Mobile Service Usage through Physical Mobile Interaction**. Washington, DC, USA: In: Proceedings of the Fifth IEEE International Conference on Pervasive Computing and Communications, 2007. 262-271p.

BUBLITZ, F. M. **Front-Frame-based Ontology System**: Uma Ferramenta para Criação e Edição de Ontologias. Universidade Federal de Alagoas: [s.n.], 2005.

CALVI C. Z., P. R. M. P. F. J. G. **Um interpretador de contexto para plataformas de serviços context-aware.** São Leopoldo/RS: Anais SEMISH (CSBC), 2005.

CHEN G., K. D. **A Survey of Context-Aware Mobile Computing Research**. Dartmouth College: Department of Computer Science, 2000.

CHEN, H. **An Intelligent Broker Architecture for Pervasive Context-Aware Systems**. University of Maryland, Baltimore: [s.n.], 2004. 121p.

CHLAMTAC I., C. M. L. J. **Mobile Ad hoc Networking**: Imperatives and Challenges. [S.l.]: Ad Hoc Networks, 2003. 13-64p.

DEY, A. K. **Providing Architectural Support for Building Context-Aware Applications.** [S.l.]: Georgia Institute of Technology, 2000.

EDWARDS. Discovery Systems in Ubiquitous Computing. **IEEE Pervasive Computing**, [S.l.], v.5, n.70-77, 2006.

FREITAS, F. **Ontologias e a Web Semântica**. Anais do XXIII Congresso da Sociedade Brasileira de Computação: [s.n.], 2003. 1-52p.

GRIMM R.; DAVIS J.; LEMAR, E. M. A. S. S. A. T. B. B. B. G. G. S. W. D. System support for pervasive applications. **ACM Trans. Comput. Syst.**, [S.l.], v.22, n.421-486, 2004.

GRUBER, T. R. **A Translation Approach to Portable Ontologies**. Knowledge Acquisition: [s.n.], 1993. 199-220p.

HARIHAR K.; KURKOVSKY, S. **Using Jini to enable pervasive computing environments**. New York, USA: In: ACM-SE 43: Proceedings of the 43rd annual Southeast regional conference, 2005. 188-193p.

HELAL, S. The Gator Tech Smart House: A Programmable Pervasive Space Computer. **IEEE Computer Society Press**, [S.l.], v.38, n.50-60, 2005.

HENRICKSEN K; INDULSKA, J. R. A. **Modeling context information in pervasive computing systems**. Zurich, Switzerland: PROCEEDINGS OF THE FIRST INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING, 2002. 167-180p.

J., K. S. K. J. M. J. P. **Visualization of hand gestures for pervasive computing environments**. Venezia, Italy: In: Proceedings of the working conference on Advanced visual interfaces, 2006.

JARDIM, C. H. O. **Context Daemon**: um serviço de notificação de eventos para aplicações cientes de contexto. [S.l.]: http://coweb.icmc.usp.br/coweb/upload/5/ProjetoGradII-Patrao-revisado.pdf, 2003. 40p.

JINI. **JINI Homepage**. [S.l.]: http://www.jini.org, 2008.

KRAUSE A.; SMAILAGIC, A. S. D. Context-Aware Mobile Computing: Learning Context-Dependent Personal Preferences from a Wearable Sensor Array. **EEE Transactions on Mobile Computing (IEEE Computer Society)**, [S.l.], v.5(2), n.113-127, 2006.

LI DING PRANAM KOLARI, Z. D. S. A. T. F. J. **Using Ontologies in the SemanticWeb**: A Survey. UMBC: [s.n.], 2005.

MCILRAITH S.; MARTIN, D. **Bringing Semantics to Web Services**. IEEE Intelligent Systems: In: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, 2003. 90-93p.

MIZOGUCHI, R. **Tutorial on Ontologival Engineering**: Advanced Course of Ontological Engineering. New Gen. Computers: [s.n.], 2004. 198-220p.

MOKHTAR S.; FOURNIER, D. G. N. I. V. **Context-Aware Service Composition in Pervasive Computing Environments**. Springer Berlin: In: Proceedings of the 2nd International Workshop on Rapid Integration of Software Engineering techniques, 2005. 129-144p.

MOSTEFAOUI G. K., R. J. P. B. P. **Context-Aware Computing**: A Guide for the Pervasive Computing Community. Beirute, Libano: Proceedings of the 2004 IEEE/ACS International COnference on Pervasive Services, 2004.

NATALYA FRIDMAN NOY RAY W. FERGERSON, M. A. M. **The Knowledge Model of Protege-2000**: Combining Interoperability and Flexibility. In Proceedings of the 12th European Workshop on Knowledge Acquisition Modeling and Mannagement: [s.n.], 2000. 17-32p.

OSGI. **OSGI Alliance Homepage**. [S.l.]: http://www.osgi.org, 2008.

PEREIRA FILHO J. G.; PESSOA, R. M. C. C. Z. O. N. Q. C. R. R. M. B. A. C. P. F. C. R. G. L. M. M. **Infraware**: um Middleware de Suporte a Aplicações Móveis Sensíveis ao Contexto. [S.l.]: In: SBRC - SIMPÓSIO BRASILEIRO DE REDES DE COMPUTADORES, 2006.

PERKINS, C. E. **Mobile IP**. [S.l.]: IEEE Communications, 1997. 66-81p.

ROBERT NECHES RICHARD FIKES, T. F. T. G. R. P. **Enabling Technology for Knowledge Sharing**. [S.l.: s.n.], 1991. 36-56p.

RUDI STUDER V. RICHARD BENJAMINS, D. F. **Knowledge Engineering**: Principles and Methods. [S.l.: s.n.], 1998. 161p.

SACRAMENTO, V.; ENDLER, M.; RUBINSZTEJN, H. K.; LIMA, L. S.; GONCALVES, K.; NASCIMENTO, F. N.; BUENO, G. A. MoCA: A Middleware for Developing Collaborative Applications for Mobile Users. **IEEE Distributed Systems Online**, Los Alamitos, CA, USA, v.5, n.10, 2004.

SATYANARAYANAN. M. Pervasive computing: vision and challenges. **Personal Communications**, [S.l.], v.8, n.10-17, 2001.

SCHILIT, B. **A Context-Aware Systems Architecture for Mobile Distributed Computing**. Columbia University: Ph.D. Thesis, 1995.

SCHILIT B.N., A. N. W. R. **Context-aware computing applications**. Santa Cruz, California: In Proceedings of IEEE Workshop on Mobile Computing Systems and Applications, 1994. 85-90p.

SOWA, J. **Semantic Networks**. In Encyclopedia of Artificial Intelligence: [s.n.], 1992.

STRANG T; LINNHOFF-POPIEN, C. **A context modeling survey**. Nottingham, England: PROCEEDINGS OF THE I INTERNATIONAL CONFERENCE ON UBIQUITOUS COMPUTING, 2004. 34-41p.

SUN J. Z., S. J. **Mobility and Mobility Management**: A Conceptual Framework. [S.l.]: In Proceedings of the 10th IEEE International Conference on Networks, 2002. 205-210p.

TRUONG K. N., A. G. D.; BROTHERTON, J. A. **Who, What, When, Where, How**: Design issues of capture access applications. [S.l.]: Proceedings of the International Conference on Ubiquitous Computing, 2001.

VALENTE, A.; BREUKER, J. **Towards Principled Core Ontologies**. [S.l.: s.n.], 1996. 96p.

WALTENEGUS, D. **Dynamic Generation of Context Rules**. [S.l.]: Lecture Notes in Computer Science, 2006. 102-115p.

WASP. **WASP Project**: http://www.freeband.nl/projecten/wasp/ENindex.html. [S.l.: s.n.], 2003.

WEISER. **The Computer for the 21st century**. San Francisco, CA, USA: Scientific American, 1995. 933-940p.

YAMIN, A. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. UFRGS, Porto Alegre, RS: Tese - Doutorado em Ciência da Computação, 2004.

ZHOU Y.; CAO, J. R. V. S. J. L. J. **A Middleware Support for Agent-Based Application Mobility in Pervasive Environments**. Washington, DC, USA: In: Proceedings of the 27th International Conference on Distributed Computing Systems Workshops, 2007. 9p.