UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

UMA ARQUITETURA PARA A UTILIZAÇÃO DE COMPUTAÇÃO NAS NUVENS NOS AMBIENTES DE COMPUTAÇÃO PERVASIVA

DISSERTAÇÃO DE MESTRADO

Henrique Gabriel Gularte Pereira

Santa Maria, RS, Brasil

2012

UMA ARQUITETURA PARA A UTILIZAÇÃO DE COMPUTAÇÃO NAS NUVENS NOS AMBIENTES DE COMPUTAÇÃO PERVASIVA

por

Henrique Gabriel Gularte Pereira

Dissertação apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para a obtenção do grau de

Mestre em Computação

Orientador: Prof. Dr. Giovani Rubert Librelotto (UFSM)

Santa Maria, RS, Brasil

2012

Universidade Federal de Santa Maria
Centro de Tecnologia
Programa de Pós-Graduação em Informática

A Comissão Examinadora, abaixo assinada,
aprova a Dissertação de Mestrado

UMA ARQUITETURA PARA A UTILIZAÇÃO DE COMPUTAÇÃO NAS NUVENS NOS AMBIENTES DE COMPUTAÇÃO PERVASIVA

elaborada por
Henrique Gabriel Gularte Pereira

como requisito parcial para obtenção do grau de
Mestre em Computação

COMISSÃO EXAMINADORA:

Giovani Rubert Librelotto (UFSM), Dr.
(Presidente/Orientador)

Cristiano André da Costa, Prof. Dr. (UNISINOS)

Juliana Kaizer Vizzotto, Profª. Drª. (UFSM)

Santa Maria, 22 de Março de 2012.

”Ash nazg durbatulûk, ash nazg gimbatul,
Ash nazg thrakatulûk agh burzum-ishi krimpatul.”
— SAURON

RESUMO

Dissertação de Mestrado
Programa de Pós-Graduação em Informática
Universidade Federal de Santa Maria

UMA ARQUITETURA PARA A UTILIZAÇÃO DE COMPUTAÇÃO NAS NUVENS NOS AMBIENTES DE COMPUTAÇÃO PERVASIVA
AUTOR: HENRIQUE GABRIEL GULARTE PEREIRA
ORIENTADOR: GIOVANI RUBERT LIBRELOTTO (UFSM)
Local da Defesa e Data: Santa Maria, 22 de Março de 2012.

O mundo atual é caracterizado pela rápida proliferação de dispositivos móveis e pelo intenso uso de computadores no nosso cotidiano. Tanto a computação pervasiva quanto a computação em nuvem têm surgido como uma tendência muito promissora. Porém, para que a computação pervasiva se consolide são necessárias algumas mudanças de paradigma nos ambientes atuais da computação. Boa parte dos problemas encontrados hoje em dia na computação pervasiva não são de ordem técnica, mas sim a falta de padrões e modelos para permitir a interoperabilidade entre os dispositivos e a criação de ambientes computacionais de baixo custo. Os ambientes de computação pervasiva são caracterizados por mudanças rápidas e frequentes, sendo necessária a existência de alguma maneira para gerenciar essa informação de contexto. Essa dissertação visa apresentar uma solução para permitir a criação de ambientes de computação pervasiva utilizando serviços disponíveis no paradigma da computação em nuvem levando em consideração requisitos como a capacidade de trabalhar com dispositivos computacionais heterogêneos consumindo o mínimo possível de recursos e utilizando ontologias para a representação de informação de contexto. Nesse contexto, são apresentadas uma proposta de arquitetura para ambientes pervasivos, um estudo de caso em um cenário residencial e apresentados resultados e conclusões sobre a arquitetura proposta. Os resultados alcançados no estudo de caso permitiram a implementação de um ambiente pervasivo utilizando recursos computacionais disponíveis na nuvem e atingind os objetivos propostos no trabalho.

Palavras-chave: Computação Ubíqua, Computação Pervasiva, Ontologias, Computação na Nuvem.

ABSTRACT

Master's Dissertation
Program in Computer Science
Universidade Federal de Santa Maria

AN ARCHITECTURE FOR THE USE OF CLOUD COMPUTING IN PERVASIVE COMPUTING ENVIRONMENTS

AUTHOR: HENRIQUE GABRIEL GULARTE PEREIRA
ADVISOR: GIOVANI RUBERT LIBRELOTTO (UFSM)
Defense Place and Date: Santa Maria, March 22nd, 2012.

The modern world can be characterized by the quick proliferation of mobile devices and by the intense use of computers on our daily lives. Both pervasive computing and cloud computing have appeared as very promissing trends, but for pervasive computing to reach mainstream, many paradigm changes are needed on the current computing environments. Some of the problems found in pervasive camputing are not from a technical order, but due to a lack of standards and models to allow devices to interoperate and the problems related to the creation of low cost computing environments. Pervasive environments are marked by having sudden and frequent changes, making it necessary to think of a way to manage context information. This work aims at showing a solution that will allow the creation of pervasive computing environments using resources available in the cloud computing paradigm and taking in consideration requisites like the ability of mixing heterogenous computing devices running on the least possible amount of resources and using ontologies for context information representation and management. In this context, an architecture for the development of pervasive computing environments, an study case in a residencial cenario and an analysis of the results obtained with the proposed architecture are presented.

Keywords: Ubiquitous Computing, Pervasive Computing, Middleware, Ontologies, Cloud Computing.

LISTA DE FIGURAS

LISTA DE CÓDIGOS FONTES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CoBrA	*Context Broker Architecture*
CONON	*CONtext ONtology*
DBaaS	*DataBase as a Service*
EC2	*Elastic Cloud Compute*
HuaaS	*Humans as a Service*
IaaS	*Infrastructure as a Service*
JSON	*Javascript Serialized Object Notation*
MA	Módulo de Atuação
MML	Módulo de Monitoramento Local
MMR	Módulo de Monitoramento Remoto
MOR	Módulo de Ontologias e *Reasoning*
OIL	*Ontology Inference Layer*
OWL	*Web Ontology Language*
PaaS	*Platform as a Service*
PDA	*Personal Digital Assistant*
PEP	Prontuário Eletrônico do Paciente
RDF	*Resource Descripton Framework*
RDFS	*Resource Descripton Framework Schema*
Saas	*Software as a Service*
SOCAM	*Service-Oriented Context-Aware Middleware*
XML	*Extensible Markup Language*
UbiComp	*Ubiquitous Computing*
SPARQL	*SPARQL Protocol and RDF Query Language*

SUMÁRIO

1 INTRODUÇÃO

Uma das características da era digital é a proliferação de dispositivos móveis heterogêneos, a popularização de sensores e habilidade desses dispositivos de coexistir tanto com as redes sem fio quanto com as redes cabeadas e a infraestrutura computacional já existente (LYONS et al., 2009). Essa proliferação vai ao encontro da visão de WEISER (1991) sobre a computação ubíqua: computadores baratos e com baixo consumo de energia, interconectados por uma rede.

Uma das principais áreas de pesquisa de computação nos últimos anos é a exploração de novas maneiras para realizar a interação humano-computador com o mundo físico em que nós vivemos (DOURISH, 2004). Avanços na tecnologia de redes e comunicações *wireless*, miniaturização de sistemas e melhorias nas interfaces ao usuário permitem uma melhor pesquisa e desenvolvimento na computação pervasiva (CHENG; MARSIC, 2002).

Muitas das tarefas realizadas no dia a dia podem ser melhoradas com a utilização de computadores (WANT; PERING, 2005) e alguns dos preceitos da computação pervasiva são auxiliar os usuários na automatização de muitas dessas tarefas e permitir aos usuários a configuração seu ambiente da maneira que mais lhes agrada (ROBINSON; WAKEMAN; OWEN, 2004).

A Computação em Nuvem, ou *Cloud Computing*, tem sido vista por alguns como a próxima mudança de paradigma (AGRAWAL; DAS; EL ABBADI, 2011; SUMTER, 2010). Ela foi capaz de causar um impacto no modelo computacional de empresas que pode ser comparado com a transição de *mainframes* para computadores pessoais (TRAN et al., 2011).

1.1 Problema

Nos anos 1990, a maior parte dos desafios na computação pervasiva era de ordem técnica (WANT; PERING, 2005). A não existência de processadores rápidos e com baixo consumo de energia, falta de aplicações pervasivas e a falta de padronização nas redes de computadores impedia que a computação pervasiva viesse a se tornar realidade. A fonte das maiores dificuldades para a implementação de um ambiente de computação pervasiva não é de ordem técnica, mas de ordem estrutural (O'SULLIVAN; LEWIS, 2003).

JAIN; WULLERT II (2002) também aponta como desafios existentes na computação pervasiva a grande quantidade de poder computacional necessário para integrar esses dispositivos, o alto consumo de energia e a dificuldade em realizar melhorias no hardware para permitir que esses ambientes atendam a demanda crescente.

São necessárias a criação de metodologias e padrões para permitir a interoperabilidade entre os dispositivos computacionais a fim de permitir que a atual infraestrutura computacional acomode a computação proativa e se torne, de certa maneira, transparente ao usuário. A utilização da computação nas nuvens pode resolver o problema do grande poder computacional necessário para o funcionamento de um ambiente de computação pervasiva, bem como permitir a realização de melhorias no hardware para atender a demanda sem que o ambiente pervasivo sofra problemas.

1.2 Objetivo Geral

O presente trabalho propõe a criação de uma arquitetura de computação pervasiva *genérica* capaz de utilizar recursos da computação em nuvem quando necessário, permitindo a criação de ambientes de computação pervasiva através da integração entre dispositivos computacionais heterogêneos, reduzindo os custos de poder de processamento inerentes a computação pervasiva e possibilitando a expansão de novos dispositivos de maneira simples e transparente ao usuário da arquitetura.

1.3 Objetivos Específicos

Para a realização do trabalho serão executadas as seguintes tarefas:

- Levantamento bibliográfico relacionado a computação pervasiva, consciência de contexto e computação em nuvem;
- Estudo dos trabalhos relacionados;
- Desenvolvimento de uma proposta de arquitetura utilizando componentes da computação em nuvem;
- Comparação da arquitetura proposta com as arquiteturas relacionadas;
- Realização de estudo de caso controlado utilizando um protótipo da arquitetura proposta;
- Análise dos resultados obtidos;

1.4 Organização Textual

O texto está organizado da seguinte forma: no Capítulo 2 são detalhados conceitos de Computação Pervasiva e os aspectos relacionados a consciência de contexto. Também são apresen-

tadas algumas arquiteturas de computação pervasiva existentes.

No Capítulo 3 são apresentados detalhes sobre ontologias e sua utilização nos ambientes de computação pervasiva como maneira de representar e processar a informação sobre o contexto.

O Capítulo 4 trata da computação em nuvem. Os principais termos, modelos de implantação e modelos de serviço são detalhados ao longo desse capítulo.

A arquitetura proposta é apresentada em detalhes no Capítulo 5. O Capítulo 6 mostra o estudo de caso utilizando a arquitetura proposta, sendo o estudo dos trabalhos relacionados encontrado no Capítulo 7. Por fim, no Capítulo 8, as conclusões acerca do trabalho são apresentadas e são sugeridos trabalhos futuros.

2 COMPUTAÇÃO PERVASIVA

A visão mais aceita de computação ubíqua ou pervasiva é a proposta por WEISER (1991) que mostra um mundo onde os computadores estão inseridos de forma natural no nosso cotidiano, centenas de computadores em uma sala, cada um voltado para uma tarefa específica e interagindo uns com os outros para realizar ações em nome do usuário. A computação ubíqua e pervasiva vai além das barreiras tradicionais de como, quando e onde ocorre a interação entre o homem e a máquina. Ela promete transformar espaços físicos em espaços computacionais proativos e inteligentes, vislumbrando um mundo onde os computadores podem oferecer serviços aos usuários não importando o local nem a hora (SOLDATOS et al., 2007).

Para que essa visão se torne realidade, são necessários três componentes: computadores baratos, com baixo consumo de energia e telas convenientes, software para aplicações pervasivas e uma maneira de interligar tudo isso (WEISER, 1991). As aplicações pervasivos têm de ser proativas, isso é, descobrindo o que o usuário deseja e providenciando a ação desejada no momento correto (LOUREIRO; OLIVEIRA; ALMEIDA, 2005).

Na computação pervasiva, a informação relacionada ao ambiente é chamada de *contexto* (DEY et al., 1999) e é percebida através de sensores ou gerada e enviada por algum dispositivo. Para desenvolver aplicações nesse paradigma é necessário que essas aplicações se adaptem dinamicamente as situações de mudança no ambiente. As aplicações utilizam o contexto para se adaptar as necessidades usuário (XU et al., 2010; HENRICKSEN; INDULSKA, 2006) e são conhecidas como aplicações conscientes de contexto.

As aplicações pervasivas são vantajosas em relação as aplicações tradicionais, já que podem se adaptar a ambientes heterogêneos utilizando a informação de contexto (LOUREIRO; OLIVEIRA; ALMEIDA, 2005). Sensores cada vez menores, mais baratos e mais precisos são desenvolvidos e estão permitindo o surgimento da chamada *"computação invisível"* (KALAPRIYA et al., 2004) e aplicações que interagem com o ambiente estão se tornando mais comuns. Hoje em dia a computação está se tornando cada vez mais móvel e pervasiva (WANT; PERING, 2005).

Os termos computação pervasiva e computação ubíqua são utilizados por alguns autores para representar conceitos diferentes, enquanto outros autores defendem que eles significam a mesma coisa. No contexto desse trabalho, a segunda opinião será utilizada, logo eles serão utilizados com o mesmo significado.

2.1 Consciência de Contexto

A consciência e o gerenciamento de contexto são dois dos desafios enfrentados computação pervasiva (FOURNIER et al., 2006). Sistemas pervasivos são marcados por duas características distintas e importantes: a percepção do contexto e a execução da tarefa.

Essa percepção do contexto e a capacidade de utilizar esse contexto para realizar a execução da tarefa são o que caracteriza uma aplicação consciente de contexto. A *context-awareness* se baseia principalmente em sensores de hardware e algoritmos de processamento de sinal (SOLDATOS et al., 2007).

A consciência de contexto é importante para ambientes de computação pervasiva poderem adaptar as entidades computacionais de acordo com as necessidades do usuários e as capacidades técnicas. Em ambientes distribuídos é especialmente importante que essa informação de contexto possa ser compartilhada entre diferentes entidades computacionais para permitir interoperabilidade (AY, 2008).

Contexto engloba mais do que a localização do usuário, porque outras coisas de interesse também são móveis e sofrem mudanças. Contexto inclui iluminação, nível de ruído, conectividade, custos de comunicação, largura de banda e até mesmo a situação social (SCHILIT; HILBERT; TREVOR, 2002).

Contexto é qualquer informação que possa ser utilizada para caracterizar a situação de uma entidade (DEY; ABOWD; SALBER, 2001). O contexto como forma de representação é uma forma de informação, delineável, estável e que pode ser separada da atividade e como problema de interação é uma propriedade relacional, com escopo definido dinamicamente e produzido pela tarefa a ser executada (DOURISH, 2004).

2.2 Aplicações da Computação Pervasiva

Sistemas de computação pervasiva tem sido desenvolvidos em várias áreas. Entre elas, é possível encontrar aplicações pervasivas nas áreas de *healthcare*, anúncios, jogos. Alguns autores chegam a propor a utilização de sistemas pervasivos em colônias espaciais (KOSTAKOS, 2011), elencando as oportunidades e desafios.

2.2.1 Healthcare

Destacam-se os trabalhos de BARDRAM (2004) com o desenvolvimento de Prontuários Eletrônicos de Pacientes (PEP) capazes de se adaptar as mudanças de contexto e o protétipo de

uma cama de hospital com uma tela que pode exibir o prontuário do paciente para o médico ou agir como uma televisão normal.

GASSEN (2010) propõe uma metodologia para a descrição de contexto em ambientes hospitalares e realiza um estudo de caso sobre uma sala de medicamentos pervasiva, capaz de informar aos enfermeiros quais e quantos os medicamentos devem ser administrados aos pacientes. LIBRELOTTO et al. (2009); FREITAS (2011) também propõem trabalhos na área de *helthcare* pervasivo.

2.2.2 Anúncios

RANGANATHAN; CAMPBELL (2002) explora o conceito de anúncios direcionados em um ambiente pervasivo. Ele realiza um estudo sobre quando é o melhor momento para a exibição de um anúncio e como isso pode ser realizado em um ambiente pervasivo.

DI FERDINANDO et al. (2009) desenvolve o sistema MyAds. Ele utiliza telas em ambientes pervasivos para exibir aos usuários específicos anúncios de interesse, ou anúncios que sejam significativos para grandes grupos de usuários no sistema pervasivo.

2.2.3 Jogos

BJORK et al. (2002) começa a utilizar ambientes físicos pervasivos como tabuleiro para jogos digitais. Seu trabalho consiste na utilização de câmeras e sensores para digitalizar em tempo real as informações presentes em um tabuleiro físico para o computador.

MAGERKURTH et al. (2004) projeta um *framework* para o desenvolvimento de jogos pervasivos em uma infraestrutura de colaboração chamada Pegasus.

2.3 Sumário do Capítulo

Processadores, sensores e atuadores estão cada vez mais presentes nos produtos utilizados durante nosso dia-a-dia, mas ainda é díficil fazer com que esses dispositivos trabalhem em conjunto e de maneira transparente, alcançando os objetivos da computação pervasiva. Os problemas de conectividade e trocas de informação podem ser resolvidos com o uso de redes de alta velocidade, ou da própria Internet. Mas ainda não é possível contar com uma integração entre esses processadores, sensores, atuadores e nem com redes de alta velocidade em cada lugar. É necessário criar metodologias e padrões para permitir a interoperabilidade entre as diferentes tecnologias existentes a fim de permitir que a visão original da computação pervasiva se torne realidade.

3 ONTOLOGIAS

O termo ontologia vem do prefixo Grego *onto-* que significa "ser, aquele que éëm conjunto com o sufixo *logia* referente a "ciência, estudo, teoria". Ontologia é uma disciplina filosófica que pode ser descrita como a *ciência da existência* ou o *estudo do ser*. Aristóteles formou o conceito lógico de ontologias contendo noções como *categorias*, *subsunções* e também a distinção entre superconceito/subconceito, referida por ele como *gênero* e *subespécie* (CIMIANO, 2006). Ele também se referiu a características que podem ser utilizadas para distinguir diferentes objetos de um mesmo gênero e permitir a classificação formal em categorias, levando a criação de subespécies. Esse é o principio no qual se baseiam as notações modernas do conceito ontológico. Aristóteles também veio a introduzir um conjunto de regras de inferência, chamados *silogismos*, que continuam ser utilizados em sistemas modernos de raciocínio lógico (SOWA, 2000).

Na computação moderna, o termo ontologia não é mais utilizado para descrever a ciência da existência, mas especificações formais de conceitualização (GRUBER, 1995). Embora originalmente o termo "ontologiaëja utilizado para representar uma ciência específica, na computação o termo *ontologia* tem surgido para representar o modelo conceitual por baixo de um certo domínio do conhecimento, descrevendo-o em uma maneira declarativa e portanto conseguindo separar-se dos aspectos procedurais desse domínio.

Ontologia pode ser utilizada para descrever vários artefatos com diferentes estruturas, indo de taxonomias simples, esquemas de metadados, até teorias lógicas (HEFFLIN; VOLZ; DALE, 2002). Na visão de LIBRELOTTO et al. (2009), "ontologias podem ser vistas como um conjunto coerente de coleções estruturadas de informação".

Os principais usos de ontologias tem sido: comunicação (entre sistemas computacionais, entre humanos, entre humanos e sistemas computacionais que necessitem trocar informações de domínio (HEFFLIN; VOLZ; DALE, 2002)), inferência computacional (representação e manipulação de planos e planejamento de informações) e para o reuso do conhecimento (organizar e estrutura informações de domínio) (GRUNINGER; LEE, 2002). Também é possível encontrar aplicações práticas na área de cybersegurança (TAKAHASHI; KADOBAYASHI; FUJIWARA, 2010), descoberta de serviços em computação de alta performance (PHAM et al., 2010), modelagem de contexto (AY, 2008; WANG et al., 2004), aplicações pervasivas de *healthcare* (GASSEN, 2010; FREITAS, 2011).

Embora o número de aplicações para a utilização de ontologias na ciência da computação esteja crescendo constantemente, é necessário que uma definição clara e formal de ontologia apareça. É possível representar uma ontologia através da utilização de um modelo matemático complexo (BOZSAK et al., 2002; CIMIANO, 2006).

As ontologias são classificadas de acordo com seu nível de generalidade, podendo ser: Ontologias Genéricas, Ontologias de Domínio e Ontologias de Aplicação (BRUIJN; FRANCONI; TESSARIS, 2005). As ontologias genéricas descrevem conceitos gerais, sem nenhuma particularidade relacionada a algum domínio em particular. Ontologias de Domínio descrevem conceitos específicos de um determinado domínio. E Ontologias de Aplicação descrevem conceitos necessários para aplicações específicas.

3.1 Definição Formal

Para BOZSAK et al. (2002), uma ontologia pode ser definida matematicamente como uma estrutura no formato:

$$\mathbb{O} := (C, \leq c, R, \sigma_R, \leq_R, A, \sigma_A, T)$$

consistindo de:

* quatro conjuntos distintos *C, R, A e T*. *C* representando os identificadores de conceitos, *R* os relacionamentos, *A* os atributos e *T* os tipos de dados;

* um retículado semi-superior (*semi-upper lattice*) $\leq_C$ em *C* com base no elemento $Raiz_C$, chamada de taxonomia;

* uma função $\sigma_R : R \rightarrow C^+$ chamada assinatura de relacionamento,

* uma ordem parcial $\leq_R \ emR$, chamada hierarquia de relacionamento, onde $r_1 \leq_R r_2$ implicando em $|\sigma R(r_1)| = |\sigma R(r_2)|$ e $\pi_i(\sigma_R(r_1)) \leq_C \pi_i(\sigma_R(r_2))$, para cada $1 \leq i \leq |\sigma R(r_1)|$;

* uma função $\sigma_A : A \rightarrow C\chi T$, chamada de assinatura de atributo;

* um conjunto *T* de tipos de dados, tais como strings, inteiros, datas, etc.

Deste modo, $\pi_i(t)$, é o n-ésimo componente da tupla *t*. Para que isso ocorra é necessário que o retículado semi-superior $\leq$ atenda as seguintes condições:

$$\forall x \ (x \leq x) \text{ (relação reflexiva)} \tag{3.1}$$

$$\forall x \forall y \ (x \leq y \wedge y \leq x \rightarrow x = y) \text{ (relação anti-simétrica)} \tag{3.2}$$

$$\forall x \forall y \forall z \ (x \leq y \wedge y \leq z \rightarrow x \leq z) \text{ (relação transitiva)} \tag{3.3}$$

$$\forall x \ (x \leq topo) \text{ (elemento superior)} \tag{3.4}$$

$$\forall x \forall y \exists z \ (z \geq x \wedge z \geq y \wedge \forall w (w \geq x \wedge w \geq y \rightarrow w \geq z)) \text{ (supremum)} \tag{3.5}$$

CIMIANO (2006) completa a definição dando os conceitos matemático de: domínio e alcance, sistema de axiomas, lexicon, base do conhecimento, lexicon de instância, extensão, indução de hierarquia conceitual, refinamento e extensão léxica.

3.2 Linguagens de Definição de Ontologias

Outra maneira de representar ontologias é utilizando linguagens de descrição de ontologias, com sintaxe bem definida e uma semântica formal, que deve ser detalhada, precisa, consistente permitir que distinções sejam feitas entre as classes, propriedades e relacionamentos (HEFFLIN; VOLZ; DALE, 2002). Logo, se a ontologia possuir identificadores apropriados, um humano conseguirá entender seu significado, bem como um programa poderá assumir a semântica de um relacionamento e atuar sistematicamente através dela (GASSEN, 2010).

Essas linguagens são muito utilizadas no contexto da Web Semântica. Se destacam o *Resource Description Framework Schema* (RDFS) (BRICKLEY; GUHA, 1999), *Ontology Inference Layer* (OIL) (FENSEL et al., 2001), *Web Ontology Language* (OWL) (DEAN; SCHREIBER, 2004) e OWL 2 (BOCK et al., 2009).

3.2.1 RDF e RDFS

O *Resource Description Framework*, ou RDF, é uma linguagem de propósito geral para a representação de informação na Internet. No RDF as propriedades podem ser mapeadas como

os atributos de um recurso e correspondem a tuplas tradicionais no formato atributo-valor. O RDF também é capaz de representar relacionamentos entre os recursos. Infelizmente o RDF não implementa a capacidade de descrever essas propriedades nem permite a descrição dos relacionamentos (BRICKLEY; GUHA, 2004).

Para suprir essas deficiência foi criado o RDF Schema (RDFS), uma extensão semântica do RDF, capaz de descrever grupos de recursos, hierarquias de classes e as relações entre esses recursos. Tais grupos de recursos podem ser divididos em *classes* e cada membro de uma classe é chamado de uma *instância* da classe.

3.2.2 OIL

Com a base formada pela utilização de tecnologias como o XML e o RDFS, FENSEL et al. (2001) desenvolveu uma linguagem chamada Ontology Inference Layer cujas principais vantagens sobre o RDF e o RDFS eram uma maior quantidade de primitivas de modelagem e maneiras mais ricas para a definição de conceitos e atributos, uma definição de semântica formal e a criação de motores de inferência capazes de trabalhar com a OIL.

O OIL possui uma arquitetura dividida em camadas, onde cada camada adicionada fornece mais funcionalidade e complexidade para a camada anterior. Isso permite que aplicações possam ser desenvolvidas para trabalhar em camadas que não necessitem tanta complexidade ou expressividade. A Figura 3.1 apresenta o modelo das camadas do OIL (FENSEL et al., 2001).

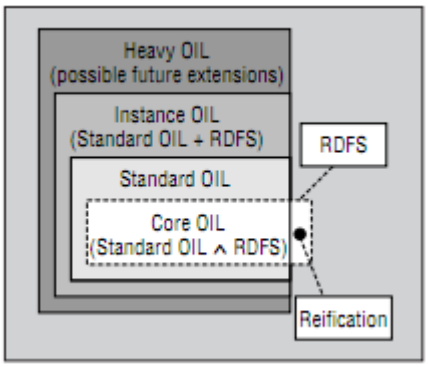

Figura 3.1: Modelo de Camadas do OIL (FENSEL et al., 2001)

A *Standard OIL* fornece as primitivas de modelagem, especificando a semântica e tornando possível a inferenciação. A camada *Instance OIL* permite a integração individual de instâncias, permitindo que o OIL trabalhe como um banco de dados, sendo capaz de armazenar várias instâncias de uma classe. E no topo fica a *Heavy OIL* onde podem ser adicionadas extensões que permitem uma maior descrição ontológica e capacidades de inferência.

3.2.3 OWL

A OWL *Web Ontology Language* foi projetada para ser utilizada por aplicações que necessitam processar informações sobre ontologias, e não só apresentar essa informação ao usuário (MCGUINNESS; HARMELEN, 2006). A OWL possui classes, relacionamentos, indivíduos e propriedades, sendo compatível com o RDF, já que a própria OWL é definida utilizando uma semântica baseada no RDF BOCK et al. (2009).

A OWL possui três subdivisões caracterizadas pelo aumento do nível de expressividade: OWL Lite, OWL DL e OWL Full. A OWL Lite é indicada para ontologias que necessitem de uma hierarquia de classificação e restrições simples. A OWL DL permite a utilização de lógica descritiva enquanto garante que a ontologia vai ser computacionalmente completa. O OWL Full permite um nível máximo de expressividade e liberdade sintática porém não possui nenhuma garantia computacional (DEAN; SCHREIBER, 2004).

3.3 Ontologias e a Computação Pervasiva

Nos últimos anos, vários sistemas de computação ubíqua e pervasiva têm utilizado ontologias: CoBrA (CHEN; FININ; JOSHI, 2004), Gaia (RANGANATHAN et al., 2004), SOCAM (PAGANELLI; BIANCHI; GIULI, 2007). A arquitetura CoBrA será apresentada no Capítulo 7 juntamente com outros trabalhos relacionados.

3.3.1 Gaia

Gaia é uma infraestrutura para ambientes inteligentes através da computação pervasiva. A principal idéia do Gaia é integrar dispositivos com o ambiente físico, permitindo que as entidades virtuais e físicas interajam de maneira satisfatória (MILLER; MCBURNEY, 2007).

Ontologias são utilizadas no Gaia para controlar a diversidade e complexidade dos recursos disponíveis no ambiente, provendo uma taxonomia padrão para os diferentes tipos de entidades e permitindo a descoberta semântica e interoperabilidade entre as entidades (RANGANATHAN et al., 2004). As ontologias são modeladas utilizando DAML+OIL.

3.3.2 SOCAM

A arquitetura *Service-Oriented Context-Aware Middleware*, ou SOCAM, permite a rápida prototipação de aplicações conscientes de contexto em ambientes pervasivos (PAGANELLI; BIANCHI; GIULI, 2007). Essa arquitetura utiliza uma ontologia genérica chamada de Con-

text Ontology (CONON) e é completado por ontologias de domínio. Todas implementadas utilizando OWL.

3.3.3 Comparativo

A Tabela 3.1 apresenta um comparativo entre as arquiteturas relacionadas que utilizam ontologias para a realização e armazenamento de informação de contexto.

Tabela 3.1: Comparativo entre as linguagens utilizadas para representar ontologias nos sistemas estudados

Sistema	**Linguagem**	**Tipo da Arquitetura**
CoBrA	OWL	Orientado a Agentes
Gaia	DAML+OIL	Mapeamento físico-virtual
SOCAM	OWL	Orientado a Serviços

3.4 Sumário do Capítulo

O uso de ontologias vai muito além da simples representação de conhecimento de um domínio. É possível realizar inferências e permitir a interoperabilidade da informação de contexto entre dispositivos diferentes dentro de um mesmo ambiente pervasivo.

As várias linguagens disponíveis para representação de ontologias acabam se tornando um obstáculo na maior aceitação da utilização de ontologias, mas isso está mudando com a popularização do OWL e a recomendação por parte da W3C. A utilização de ontologias para a representação de contexto em ambientes de computação pervasiva é uma tendência muito forte.

4 COMPUTAÇÃO EM NUVEM

MELL; GRANCE (2011) descreve como computação em nuvem como "um modelo computacional com a habilidade de permitir, de maneira ubíqua e conveniente, o acesso sob-demanda a recursos computacionais compartilhados e configuráveis". Esse modelo é composto de cinco características essenciais: habilidade de escalonar recursos sob-demanda, acesso através de uma rede, elasticidade, mensuração do uso e um *pool* de recursos. HU et al. (2011) resume computação em nuvem como um modelo computacional para compartilhamento de dados e serviços no qual está se desenvolvendo a nova geração de computação.

CUSUMANO (2010) acredita que a computação em nuvem emergiu gradativamente. A idéia de disponibilizar aplicações de software sobre uma rede é antiga, tendo surgido com o conceito de *time sharing* durante as décadas de 1960 e 1970, e evoluiu em aplicações hospedadas durante entre os anos de 1980 e 1990. Na segunda metade dos anos 1990, muitas empresas começaram a utilizar a Internet para fornecer aplicações de software. Na visão de PAN; BLEVIS (2011), a computação em nuvem é uma onda importante, com precedente histórico e que tem implicações na maneira como as pessoas interagem com alguns tipos de tecnologias digitais.

Uma funcionalidade característica da *cloud computing* é a disponibilização elástica de recursos computacionais. Usuários podem alocar e desalocar esses recursos dinamicamente a partir da necessidade da aplicação em execução (APPAVOO et al., 2010; FOUQUET; NIEDERMAYER; CARLE, 2009). O modelo de *cloud computing* enfatiza a habilidade de escalar recursos de acordo com a demanda (NAPPER; BIENTINESI, 2009). Diferentemente de sistemas tradicionais como *grids* ou *clusters*, não existe investimento inicial em infraestrutura ou pessoas e as despesas continuas são simplificadas. O custo total pode ser perto de zero quando os recursos computacionais não estão em uso e o usuário paga proporcionalmente ao seu uso. Outra vantagem da utilização da computação em nuvem é a preocupação reduzida com a perda de dados ou com a intrusão de um vírus já que os provedores da nuvem empregam hardware de armazenamento confíavel (LIANG, 2011; REESE, 2009).

Uma grande quantidade de processadores e uma infraestrutura gigantesca de comunicação são características tanto de supercomputadores quanto de sistemas de computação em nuvem. Esses dois ambientes de computação são projetados para suportar múltiplos usuários independentes que não são os donos dessas máquinas, mas utilizam uma fatia do seu poder computaci-

onal para a realização de seus trabalhos (APPAVOO et al., 2010).

Para RELLERMEYER; DULLER; ALONSO (2009) o termo *computação em nuvem* é utilizado para se referir a muitas tecnologias diferentes: da terceirização do processamento de informações até a utilização de *datacenters* externos. Mas, independente da definição utilizada, a base de software que permite isso não é nova: a *cloud computing* é geralmente implementada como uma aplicação distribuída ou paralela, executando em *clusters* virtualizados de computadores. Esse modelo sofre pela falta de metodologias adequadas para desenvolvimento e encontra muitos desafios semelhantes aos encontrados em aplicações distribuídas ou paralelas.

A computação em nuvem tem se tornado cada vez mais popular. Empresas como Amazon, Google, IBM e Microsoft oferecem serviços na nuvem. Esses serviços são executados em servidores virtualizados, hospedados dentro de seus *datacenters* e acessados pela Internet (FOUQUET; NIEDERMAYER; CARLE, 2009). Os modelos de uso e pagamento de serviços na nuvem são uma grande diferença entre o modelo de *cloud computing* e o modelo tradicional de aluguel ou compra de servidores. O usuário dos serviços na nuvem paga de acordo com a utilização dos recursos computacionais, como por exemplo horas de processamento ou volume de armazenamento utilizado. A Figura 4.1 mostra de forma organizada as características de um serviço na nuvem, os modelos de serviço e os modelos de implantação.

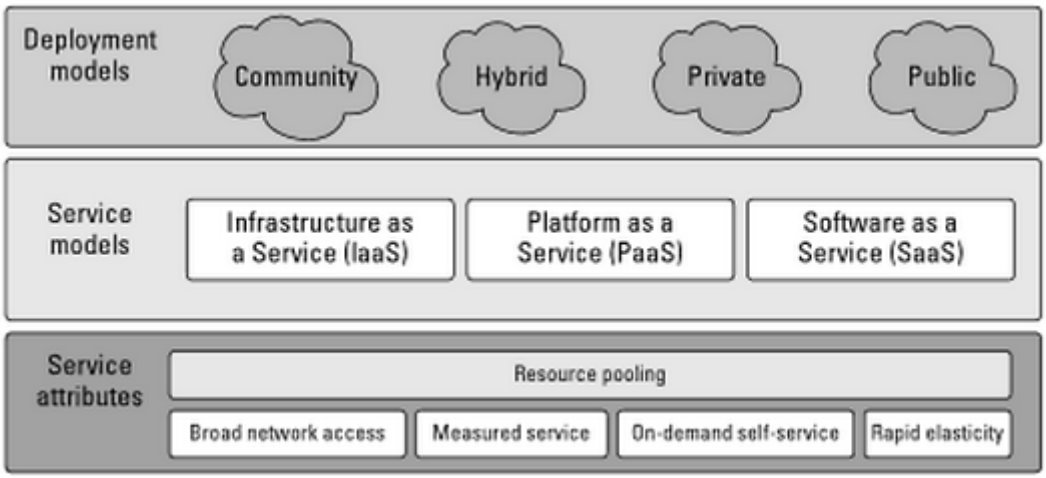

Figura 4.1: Definição visual da Nuvem Computacional (MELL; GRANCE, 2011)

4.1 Modelos de Serviço

A computação em nuvem é um termo abstrato, resultante da união de alguns conceitos como Infraestrutura como Serviço ou *Infrastructure as a Service (IaaS)*, Plataforma como Serviço ou *Platform as a Service (PaaS)* e Software como Serviço ou *Software as a Service (SaaS)*. Esse

conceito também pode ser estendido para Bancos de Dado como Serviço ou *DataBase as a Service (DBaaS)* (AGRAWAL; DAS; EL ABBADI, 2011) e Humanos como Serviço ou *Human as a Service (HuaaS)* (LENK et al., 2009). Na Figura 4.2, é possível examinar a pilha da computação em nuvem, destacando os diferentes modelos de serviço empregados em *cloud computing*.

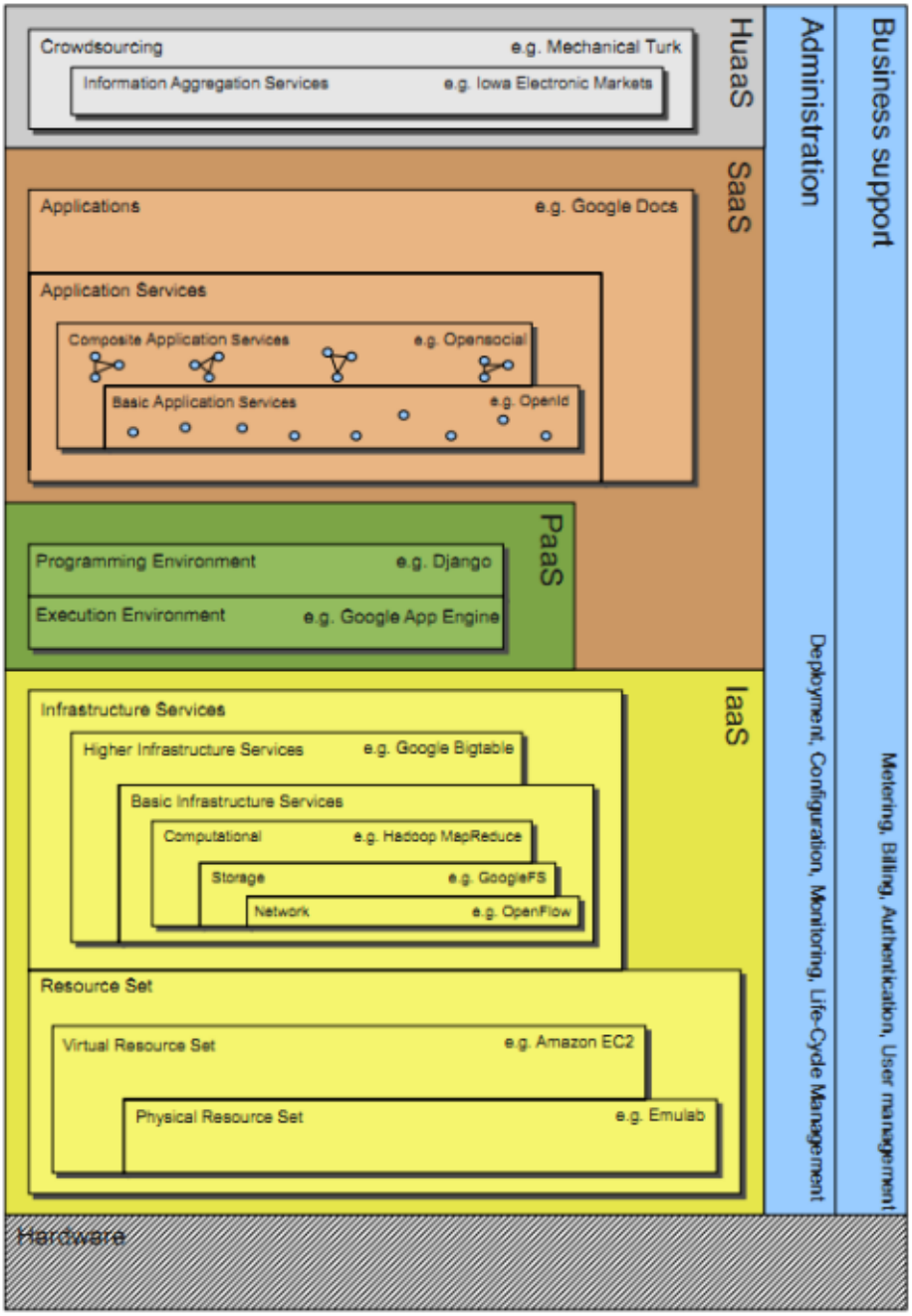

Figura 4.2: Pilha de Serviços (LENK et al., 2009)

Os três principais modelos de serviço: *IaaS*, *PaaS*, *SaaS* serão abordados a seguir:

4.1.1 Infraestrutura como Serviço

É o serviço que consiste no fornecimento de processamento, armazenamento, rede e outros recursos computacionais fundamentais onde o usuário tem a capacidade de executar e implementar qualquer software (MELL; GRANCE, 2011). O usuário não gerencia ou controla a infraestrutura da nuvem onde o serviço é prestado, mas tem controle sobre o sistema operacional, aplicações instaladas e um controle limitado sobre os componentes de rede.

Os recursos geralmente são disponibilizados através máquinas virtuais (AKHANI; CHUADHARY; SOMANI, 2011), armazenamento virtual, infraestrutura virtual, e outros ativos de hardware como recurso para o cliente, que pode requisitá-los conforme necessário SOSINSKY (2011). O provedor de serviço gerencia toda a infraestrutura, enquanto o cliente é responsável apenas pelos outros aspectos da implantação do sistema. Este modelo é vantajoso para usuários empresariais, já que não envolve investimento na construção e manutenção de *datacenters* (BHASKAR; RIMAL; LUMB, 2010).

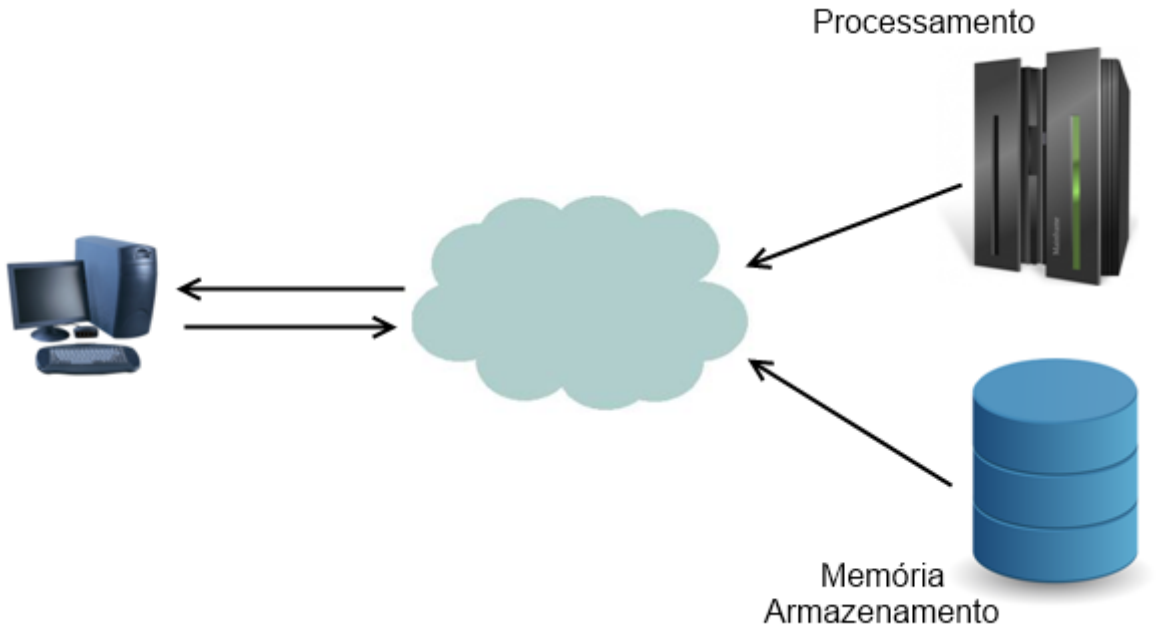

Figura 4.3: Infraestrutura como Serviço

Essa infraestrutura pode ser escalada para cima ou para baixo de acordo com as necessidades do usuário e das aplicações que estão sendo executadas (VELTE; VELTE; ELSENPETER, 2009). HAZELHURST (2008); NAPPER; BIENTINESI (2009) demonstram que é possível utilizar IaaS para estabelecer um ambiente de alta performance para computação científica.

Exemplos da utilização de Infraestrutura como Serviço podem ser vistos em Amazon EC2,

Linode, Rackspace, Flexiscale.

4.1.2 Plataforma como Serviço

É a camada da computação em nuvem que permite ao consumidor implementar e executar aplicações na infraestrutura da nuvem. Essas aplicações podem ser criadas utilizando linguagens, bibliotecas, serviços e ferramentas suportadas pelo provedor do serviço (MELL; GRANCE, 2011). O usuário não tem acesso ou controle a infraestrutura da nuvem, incluindo rede, servidores, sistema operacional ou armazenamento. Embora seja possível ajustar algumas configurações do ambiente onde a aplicação será executada.

BHASKAR; RIMAL; LUMB (2010) argumenta sobre a criação da Plataforma como Serviço dizendo que a parte que mais consome tempo e trabalho no desenvolvimento de sistemas é a criação e manutenção da infraestrutura. Por isso PaaS surge como uma alternativa para a resolução desse problema e redução dos custos de desenvolvimento e implantação.

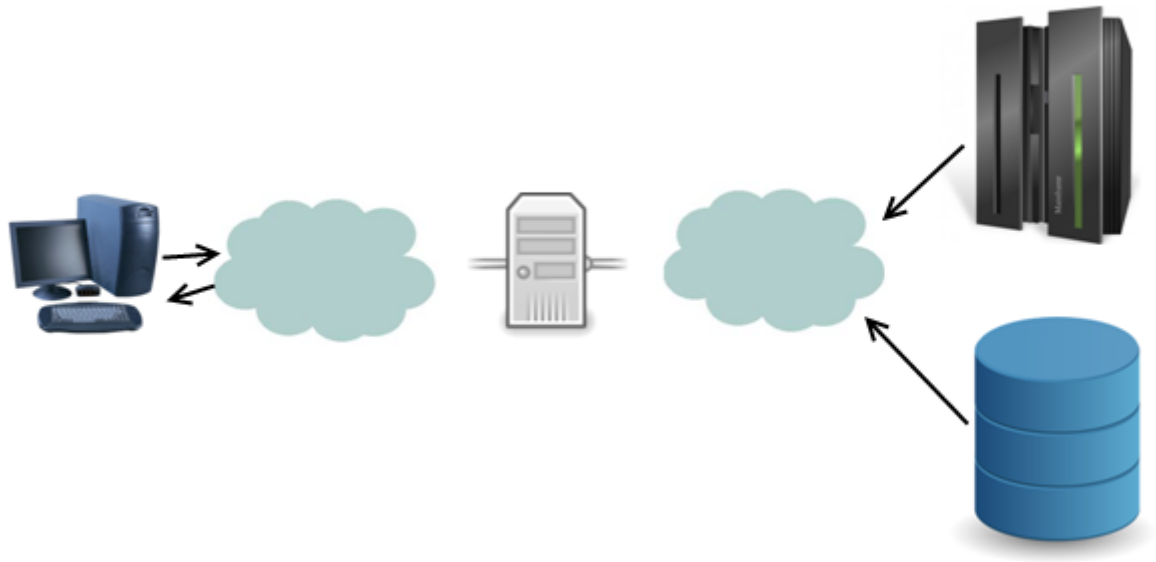

Figura 4.4: Plataforma como Serviço

LENK et al. (2009) categoriza a plataforma como serviço em dois grupos: *ambientes de programação* e *ambientes de execução*. Comparado com ambientes tradicionais de desenvolvimento de aplicações, a estratégia de utilização de PaaS pode resultar em um tempo de desenvolvimento reduzido e oferecer dezenas de ferramentas e serviços que permitem uma escalabilidade rápida da aplicação (BHASKAR; RIMAL; LUMB, 2010). Uma das maneiras mais comuns da utilização da modalidade plataforma como serviço é através de *API*s disponibiliza-

das pelo fornecedor do serviço (FOUQUET; NIEDERMAYER; CARLE, 2009; TAKAHASHI; KADOBAYASHI; FUJIWARA, 2010).

Um lado negativo da utilização de PaaS é que os provedores de serviço não permitem interoperabilidade ou portabilidade entre diferentes servidores (VELTE; VELTE; ELSENPETER, 2009). Isto é, se você desenvolver a aplicação em uma plataforma, será díficil transferir essa aplicação para outro provedor de serviço.

Exemplos da utilização de Plataforma como Serviço podem ser vistos no Google AppEngine, Microsoft Azure, Heroku.

4.1.3 Software como Serviço

Toda as as aplicações que são executadas na nuvem e fornecem acesso direto ao consumidor podem ser classificadas nesse modelo (LENK et al., 2009). As aplicações SaaS são construídas para permitir sua utilização por múltiplos usuários, de forma simultânea (VELTE; VELTE; ELSENPETER, 2009). As únicas responsabilidades do cliente são o envio e gestão dos dados que a aplicação irá processar e as etapas de interação com a aplicação. Todo o resto é de responsabilidade da empresa que fornece o serviço (SOSINSKY, 2011). Para SUMTER (2010) esse modelo de serviço pode ser resumido como um serviço que permite o aluguel do software ao usuário.

Uma das principais vantagens do SaaS é financeira, já que é mais barato para o usuário pagar somente por aquilo que ele utiliza da aplicação do que pagar o custo total de aquisição . Outras vantagens da SaaS incluem: acesso pela rede, necessidade de uma equipe de TI reduzida para realizar manutenção da aplicação, habilidade de personalização, segurança (VELTE; VELTE; ELSENPETER, 2009).

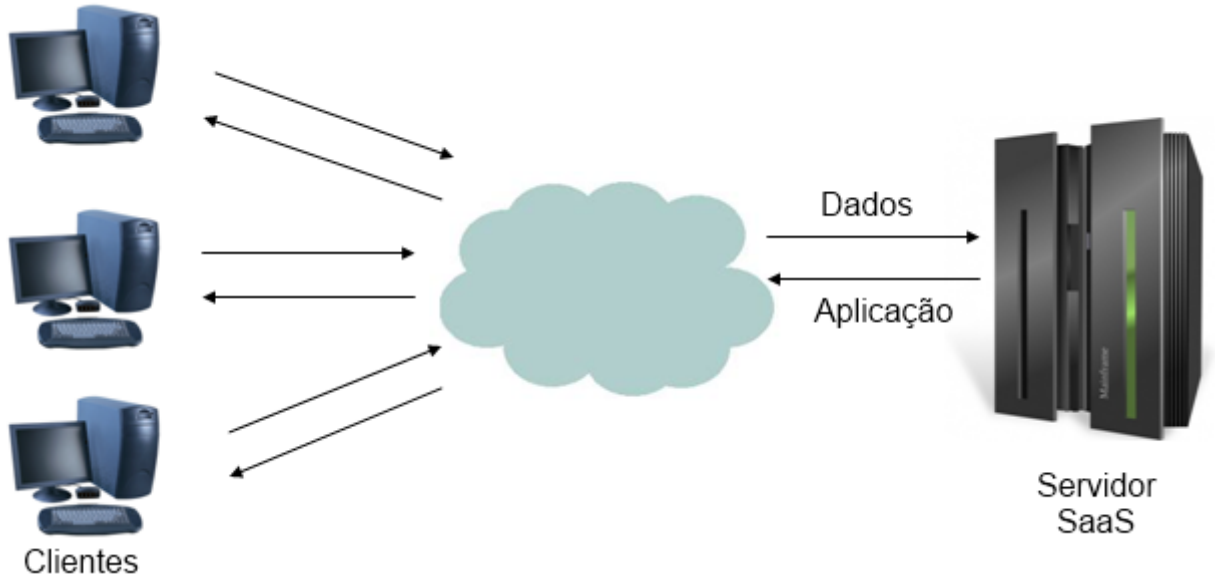

Figura 4.5: Software como Serviço

A utilização de SaaS enfrenta alguns obstáculos como a falta de aplicações para setores específicos e o *"lock-in"* dos dados da aplicação, que não permite a migração dos dados do usuário para serem utilizados em outro serviço do mesmo tipo.

Exemplos da utilização de Software como Serviço podem ser vistos em SalesForce, NetSuite, Google Docs.

4.2 Modelos de Implantação

Embora o termo *"nuvem"* seja muitas vezes utilizado para denotar o desconhecimento do local físico de onde os recursos computacionais estão sendo captados, é possível encontrar quatro modelos principais de implantação: a nuvem privada, a nuvem comunitária, a nuvem pública e a nuvem híbrida.

4.2.1 Nuvem Privada

A infraestrutura da nuvem é configurada para uso exclusivo de uma única organização, composta de múltiplos consumidores (MELL; GRANCE, 2011). Uma nuvem privada pode pertencer, ser gerenciada ou configurada pela própria organização, por uma empresa externa ou por alguma combinação entre a própria organização e empresas externas.

4.2.2 Nuvem Comunitária

A infraestrutura de nuvem é configurada para uso exclusivo de uma comunidade específica de consumidores de organizações que tenham objetivos em comum (MELL; GRANCE, 2011). Uma nuvem comunitária pode pertencer, ser gerenciada ou configurada por uma ou mais das organizações presentes na comunidade, por uma empresa externa ou por alguma combinação entre elas.

4.2.3 Nuvem Pública

A infraestrutura de nuvem é configurada para uso aberto pelo público em geral (MELL; GRANCE, 2011). Pode pertencer, ser gerenciada ou configurada por uma empresa, uma faculdade, uma entidade governamental ou por uma combinação entre elas.

4.2.4 Nuvem Híbrida

A infraestrutura de nuvem é composta de duas ou mais infraestruturas de nuvem distintas (privada, pública ou comunitária) que se mantém nuvens separadas, mas que conseguem interoperar através da utilização de tecnologias que permitem a portabilidade entre dados e aplicações (MELL; GRANCE, 2011).

4.3 Sumário do Capítulo

A computação em nuvem vem se consolidando como um bom paradigma de desenvolvimento. Os modelos de serviço contemplam desde a maior necessidade de controle de software e hardware até sistemas de mais alto nível.

O modelo software como serviço se mostra muito promissor, permitindo que usuários deixem de adquirir softwares e paguem apenas pelo seu uso. Já para desenvolvedores, o modelo de infraestrutura como serviço pode reduzir em muito os custos associados com a criação de um sistema ou plataforma. Os diferentes modos de implementação, aliados aos modelos de serviço permitem uma flexibilidade muito grande para aplicações baseadas na nuvem.

5 PROPOSTA DE ARQUITETURA PARA UTILIZAÇÃO DE COMPUTAÇÃO NAS NUVENS EM AMBIENTES DE COMPUTAÇÃO PERVASIVA

Para que um sistema se torne pervasivo é necessário que a utilização dos dispositivos computacionais aconteça de forma natural e inerente (GASSEN, 2010). Essa não é uma tarefa simples e por isso várias soluções foram propostas ao longo do tempo, soluções estas que serão discutidas no Capítulo 7. A proposta apresentada nessa dissertação visa o desenvolvimento de uma arquitetura capaz de permitir que os ambientes de computação em nuvem utilizem recursos computacionais disponíveis na nuvem de modo a permitir a construção de sistemas pervasivos capazes de satisfazer a visão de WEISER (1991).

Tradicionalmente um ambiente de computação pervasiva é composto de sensores, ligados a uma rede local que se comunicam com um concentrador. No concentrador são realizados o gerenciamento das informação de contexto, as inferências sobre o ambiente e a transmissão desses dados para os dispositivos computacionais presentes no ambiente (MEDVIDOVIC; MALEK, 2007). A Figura 7.3 apresenta a visão de uma arquitetura tradicional de computação pervasiva, com sensores conectados a *gateways* de acesso e interligados a um *backbone*, se comunicando com um concentrador central que processa as informações de contexto e interage com os dispositivos do ambiente.

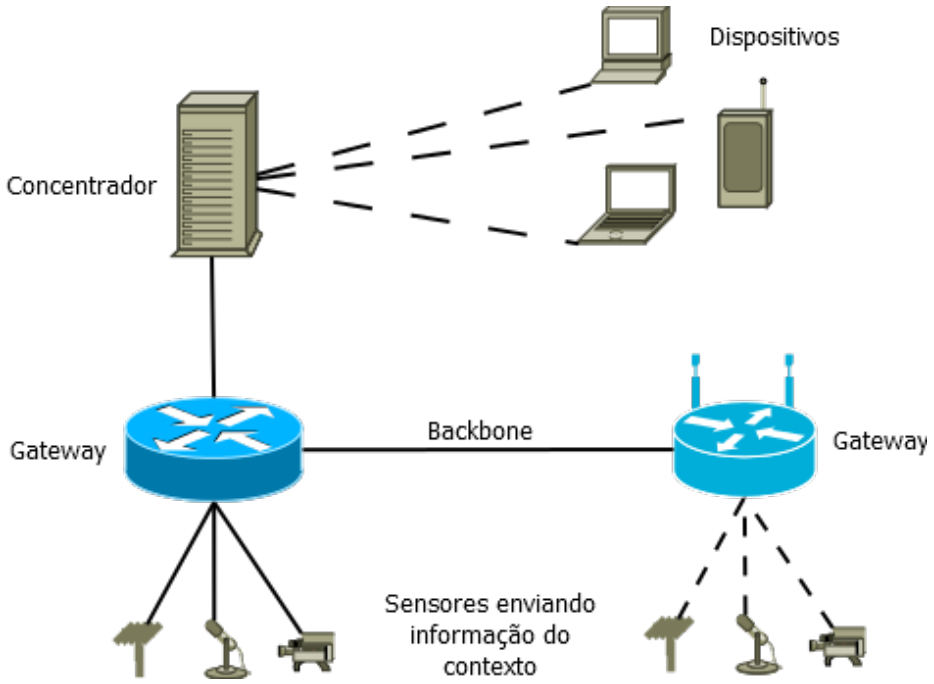

Figura 5.1: Arquitetura Pervasiva Tradicional

A proposta apresentada nessa dissertação difere das arquiteturas tradicionais por utilizar não só recursos disponíveis no ambiente de computação local, mas também recursos disponíveis em uma nuvem computacional (seja ela pública ou privada).

A arquitetura proposta é dividida em dois módulos de componentes: um módulo pervasivo local e um módulo de serviços na nuvem computacional. O módulo local é responsável pela coleta de informação de atuação sobre o ambiente pervasivo (sensores, dispositivos, informação de contexto), enquanto o módulo remoto é responsável por realizar o processamento e as inferências utilizando a informação disponibilizada pelo módulo pervasivo local. Uma visão geral da arquitetura pode ser vista na Figura 5.2.

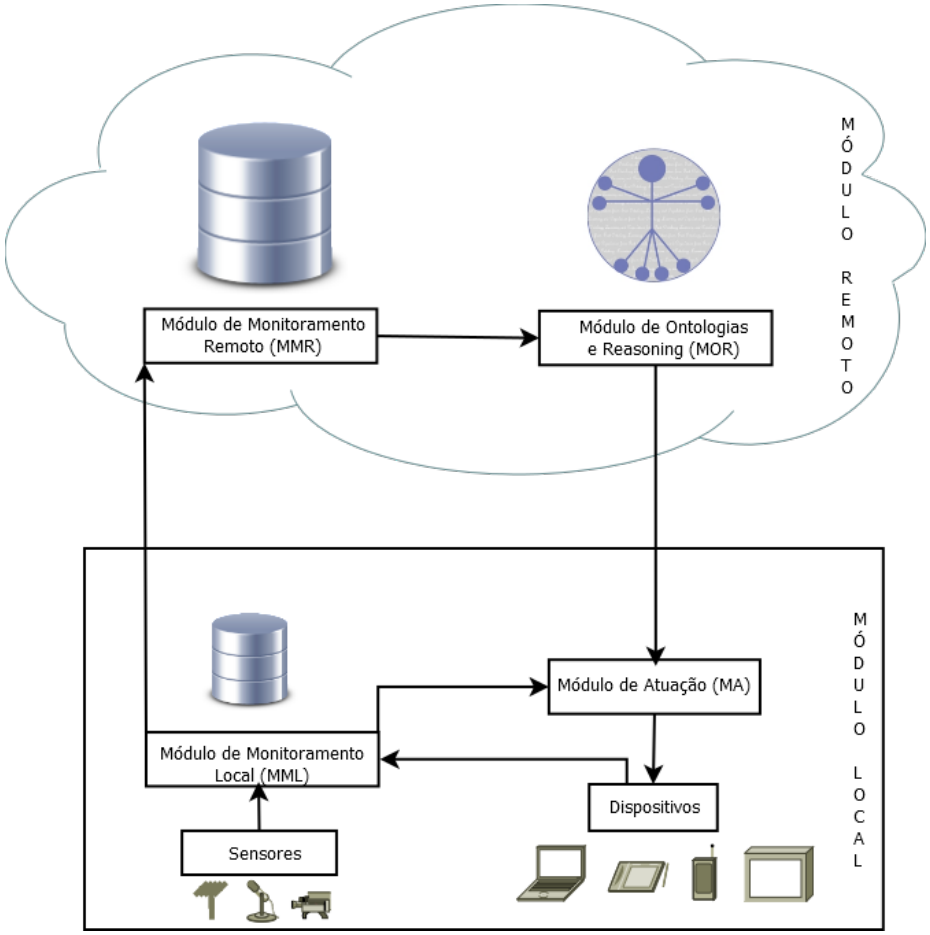

Figura 5.2: Visão Geral da Arquitetura Proposta

Esse capítulo está estruturado da seguinte maneira: primeiro serão detalhados os requisitos levados em consideração para o desenvolvimento da arquitetura, seguido por uma visão dos elementos presentes no Módulo Local e depois dos elementos constituintes do Módulo Remoto. Após o detalhamento dos componentes da arquitetura serão apresentadas as informações relativas as ontologias presentes na arquitetura, seu processamento e peculiaridades sobre o modelo

de escalabilidade que pode ser aplicado ao presente trabalho.

5.1 Requisitos

Para o desenvolvimento dessa arquitetura foram levantados uma série de requisitos funcionais e requisitos para o desenvolvimento de aplicações pervasivas para essa arquitetura. Tais requisitos são baseados nos sistemas já existentes de computação pervasiva e nas possibilidades que a computação em nuvem oferece.

5.1.1 Consumo de Recursos

A limitação no poder computacional no ambiente local é um problema que pode ser resolvido com a utilização de recursos provenientes da nuvem. Os sensores presentes no ambiente local não necessitam armazenar informações relativas a estados anteriores, sendo necessário apenas o envio de informações através da rede.

A nuvem computacional deve ser capaz de fornecer recursos como armazenamento, poder de processamento e capacidade de inferência. A comunicação entre o Módulo Local e o Módulo Remoto será realizada através da troca de mensagens utilizando uma rede de alta velocidade e alta disponibilidade.

5.1.2 Performance e Escalabilidade

O ambiente não deve ser afetado em termos de performance pelo aumento no número de dispositivos. A arquitetura deve ser flexível o bastante para suportar ambientes pervasivos variando de dezenas até centenas de dispositivos.

Esse requisito é essencial para o bom funcionamento de um ambiente de computação pervasiva já que não é possível prever a quantidade de dispositivos computacionais que estarão presentes no ambiente.

5.1.3 Heterogeneidade de Dispositivos

A atual gama de dispositivos computacionais exige que a arquitetura do sistema pervasivo seja capaz de acomodar uma quantidade muito variada de formatos de tela e sistemas operacionais já existentes e permitir que novos dispositivos ou sistemas sejam adicionados ao ambiente pervasivo sem que seja necessária a redefinição da arquitetura ou ajustes no ambiente pervasivo.

A grande variedade de tipos de tela, poder de processamento e tempo de bateria dos dispositivos vão necessitar que o ambiente se adapte de maneira adequada e transparente.

5.1.4 Heterogeneidade de Aplicações

A arquitetura deve permitir a existência de aplicações variadas no ambiente pervasivo. Não sendo limitada a uma aplicação específica e criando maneiras de desenvolver software pervasivo conforme a necessidade de cada ambiente ou dispositivo. As aplicações devem ser capazes de interoperar com o ambiente pervasivo.

A grande variedade de sistemas operacionais, tanto para dispositivos móveis (IOS, Android, BADA, S60) quanto para computadores e notebooks (Windows, Linux, OSX), não deve interferir no funcionamento do ambiente de computação pervasiva. Estratégias para atacar esse problema podem ser a utilização de aplicações específicas para cada sistema operacional, a utilização de aplicações *cross-plataform* ou o uso de aplicações hospedadas em máquinas remotas.

5.1.5 Utilização de Ontologias

Cada ambiente deverá ter uma ontologia com a descrição do contexto, dispositivos e aplicações disponíveis. A arquitetura contará com um modelo básico de ontologia que pode ser estendido para atender melhor aos anseios de cada ambiente pervasivo.

A ontologia deverá modelar o contexto de maneira adequada, levando em consideração que o ambiente pervasivo é um ambiente em constante mudanças e respeitando todos os requisitos anteriores.

5.1.6 Interoperabilidade

Deverá ser possível realizar a transferência de dados e informação de contexto entre diferentes aplicações. Todas as alterações realizadas por uma aplicação ficarão disponíveis para as outras aplicações pervasivas através do uso de ontologias. Para permitir a interação entre o ambiente local e o ambiente remoto também se faz necessária a utilização de protocolos padronizados.

Não será necessário realizar a mobilidade de aplicações, apenas a mobilidade das informações de contexto. Esse requisito visa facilitar a heterogeneidade de dispositivos e de aplicações, permitindo que uma maior quantidade de dispositivos utilize-se da infraestrutura do ambiente pervasivo.

5.1.7 Implantação

A adição de novos dispositivos ou softwares ao ambiente pervasivo deverá ser realizada através da atualização da ontologia de contexto do ambiente. Isso deverá ser realizado de maneira

fácil e se possível transparente.

Para cada novo dispositivo ou aplicação deverá ser criada uma instância da respectiva classe dentro da ontologia de contexto. Os softwares pervasivos deverão ter conhecimento prévio da ontologia de contexto e dos dispositivos presentes no ambiente.

5.2 Módulo Local

A pilha local é composta por: sensores, dispositivos e softwares capazes de executar aplicações ubíquas, o Módulo de Monitoramento Local (MML) e o Módulo de Atuação, ou Módulo Atuador (MA), interligados por uma rede ubíqua. Ela tem como objetivo reduzir o custo computacional da computação pervasiva, permitindo que o processamento pesado seja realizado na nuvem.

Os sensores são responsáveis por coletar a informação de contexto desejada. Cada sensor tem uma tarefa específica: medir a temperatura, medir o nível de ruído, verificar o status da energia, entre outros.

O MML é responsável por coletar as informações disponíveis nos diferentes sensores e dispositivos presentes no ambiente pervasivo, realizar um processamento simplificado sobre as informações de contexto e emitir ordens ao Módulo de Atuação ou disponibilizar as informações de contexto para os serviços presentes na nuvem.

O MA deve ser capaz de receber comandos tanto do Módulo de Monitoramento Local quanto do Módulo de Ontologias e *Reasoning* e enviar esses comandos para os dispositivos presentes no ambiente pervasivo local. A Figura 5.3 apresenta o Módulo Local.

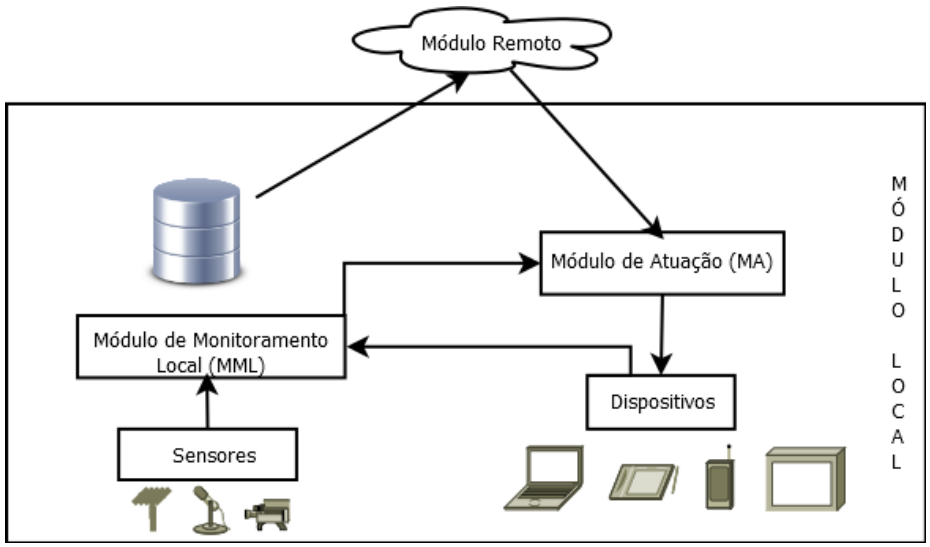

Figura 5.3: Visão do Módulo Local

5.2.1 Sensores e Módulo de Monitoramento Local

Ambientes pervasivos devem possuir sensores que monitoram o ambiente a procura de mudanças relevantes para o funcionamento do sistema pervasivo, de maneira a detectar mudanças de contexto para que sejam utilizadas em favor dos usuários (FREITAS, 2011).

A arquitetura é projetada de maneira a acomodar qualquer quantidade de sensores e não possui relação aos seus tipos. Os Sensores fornecerão a informação de contexto do ambiente pervasivo ao MML. Eles estarão ligados ao ambiente por meio de uma rede ubíqua, podendo ser cabeada ou sem-fio.

O Módulo de Monitoramento Local coleta e armazena as informações de contexto relacionadas especificamente com o ambiente que está sendo monitorado, isso é, a informação dos sensores e dispositivos que fazem parte do ambiente pervasivo local.

5.2.2 Módulo Atuador

Os atuadores poderão ser tanto físicos (*hardware*) quanto lógicos (*software*) e deverão permitir conexões externas provenientes do Módulo Remoto. Atuadores físicos poderão agir fisicamente sobre o ambiente, enquanto atuadores lógicos modificarão o software pervasivo em execução.

O MA deverá ser capaz de se comunicar com o software pervasivo presente nos dispositivos pervasivos e enviar para esses dispositivos as ações processadas pelo Módulo de Ontologias e *Reasoning*. O MA pode utilizar informações presentes no MML para localizar os dispositivos e serviços em execução.

5.3 Módulo Remoto

A pilha de serviços na nuvem é composta de dois componentes: o Módulo de Monitoramento Remoto (MMR) e o Módulo de Ontologias e *Reasoning* (MOR), interligados a uma rede ubíqua. O principal objetivo do *stack* de serviços na nuvem é realizar o processamento pesado dos ambientes pervasivos locais, aumentando ou diminuindo os recursos computacionais destinados ao MMR e ao MOR de acordo com a demanda desses ambientes.

O MMR tem como objetivo armazenar as informações enviadas pelos MMLs das camadas pervasivas locais e enviar informações relativas a alteração de contexto para o Módulo de Ontologias e *Reasoning*. Esse módulo tem de ser capaz de armazenar e gerenciar a informação relativa ao contexto de inúmeras pilhas pervasivas locais e por isso deve suportar *multitenancy*.

O processamento das informações de contexto será realizado no MOR. Cada pilha local deverá possuir uma ontologia correspondente ao ambiente armazenada nesse módulo. Assim, o MMR reportar alguma alteração de contexto, o MOR realizará o processamento dessas informações e caso seja necessário alguma ação junto ao ambiente local, enviará essa instrução diretamente para o Módulo Atuador da pilha local. A Figura 5.4 apresenta o Módulo Remoto.

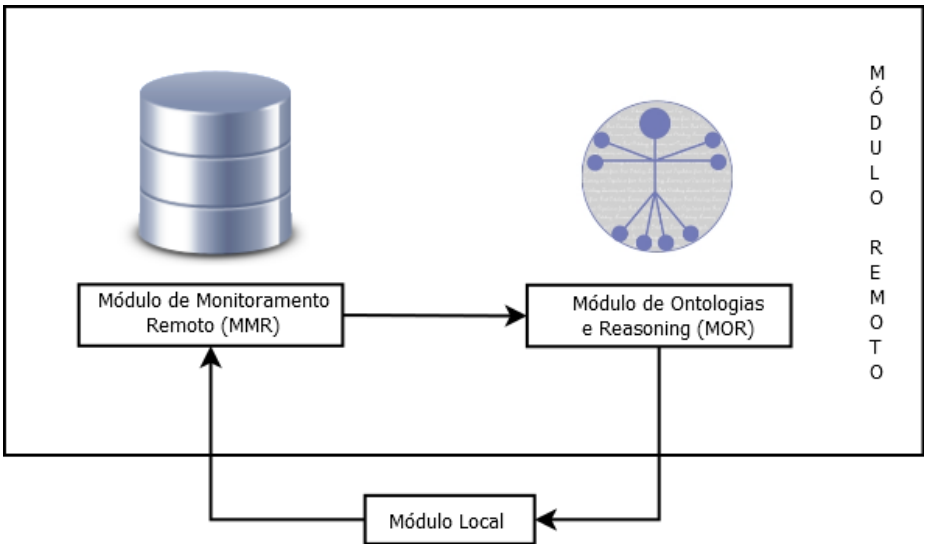

Figura 5.4: Visão do Módulo Remoto

5.3.1 Armazenamento

O MMR deverá contar com a capacidade de armazenamento de dados para permitir que a informação de contexto enviada pelo MML seja armazenada, transmitida ao MOR ou compartilhada por outros componentes do Módulo Remoto. A utilização de serviços de PaaS pode permitir que o armazenamento seja escalonado sob demanda.

A demanda de espaço para o armazenamento das informações de contexto vai variar de acordo com a quantidade de módulos de monitoramento locais conectados ao MMR e com a quantidade de sensores que estarão sendo monitorados por cada um desses MMLs.

5.3.2 Processamento

O cerne de processamento da arquitetura se dá no MOR. Todo o processamento de ontologias e mapeamento das informações de contexto presentes no MMR será executado nesse componente. A utilização de serviços de PaaS pode permitir que o poder de processamento do MOR seja aumentado ou diminuído conforme a necessidade.

O poder computacional necessário para realizar o processamento dos ambientes pervasivos locais dependerá da quantidade de dispositivos e sensores presentes nos módulos locais pervasi-

vos que são atendidos pelo MOR. Uma maior quantidade de regras e consultas sobre a ontologia de contexto desses ambientes também poderá resultar em um aumento da necessidade de poder de processamento.

5.4 Ontologia para representação de contexto na arquitetura proposta

Para realizar a representação de contexto e possibilitar o processo de inferência sobre o ambiente pervasivo, foi criada uma ontologia genérica em OWL, utilizando a ferramenta Protégé 4, que poderá ser estendida conforme a necessidade de cada sistema. A metodologia utilizada para a criação da ontologia foi a proposta por NOY; MCGUINNESS (2001).

A ontologia de contexto da arquitetura é composta por classes que representam as entidades do ambiente pervasivo (por exemplo, *Dispositivo*), propriedades que caracterizam essas classes (por exemplo, *nome, idade, altura*), e relacionamentos entre essas classes (por exemplo, *Sensor está presente em Sala*).

5.4.1 Classes

O modelo da ontologia para representação de contexto é composto por quatro classes principais: *Device, Place, Software, Person*. Essas classes foram escolhidas pois são os principais componentes de um ambiente pervasivo e vão afetar diretamente o contexto do sistema. As classes e suas respectivas descrições são exibidas na Tabela 5.1.

As classes *Software* e *Device* são abstratas, isso é, não possuem nenhum objeto e vão dar origens a outras subclasses. A classe *Software* dá origem as classes *Application* e *Operating System*. Já a classe *Device* contém as classes *Computer, Sensor, Mobile e Screen*. A Figura 5.5 apresenta o grafo com a hierarquia de classes dessa ontologia. As classes *Person e Place* podem ser estendidas conforme a necessidade do ambiente pervasivo, podendo permitir a diferenciação entre diferentes níveis de usuário (por exemplo: administrador, morador, visitante) e diferentes tipos de ambiente (por exemplo: sala de aula, quarto hospitalar, corredor).

Tabela 5.1: Descrição das classes

Classe	**Descrição**
Person	Representa uma pessoa capaz de interagir com o ambiente pervasivo e realizar alterações de contexto.
Place	Representa um local físico no ambiente pervasivo onde as ações e interações vão ser realizadas.
Device	Classe abstrata que representa um dispositivo computacional presente no ambiente.
Computer	Representa um Computador, pode se conectar a várias telas.
Sensor	Captura alguma propriedade do ambiente onde está instalado.
Mobile	Modela um dispositivo móvel capaz de realizar computações.
Screen	É uma tela que tem como principal objetivo apresentar informações para o usuário.
Software	Classe abstrata responsável por retratar os softwares presentes no ambiente pervasivo.
Application	São as aplicações que podem ser executadas por dispositivos presentes no sistema pervasivo.
Operating System	Representa o Sistema Operacional que pode ser executado pelos dispositivos do ambiente e pode executar aplicações.

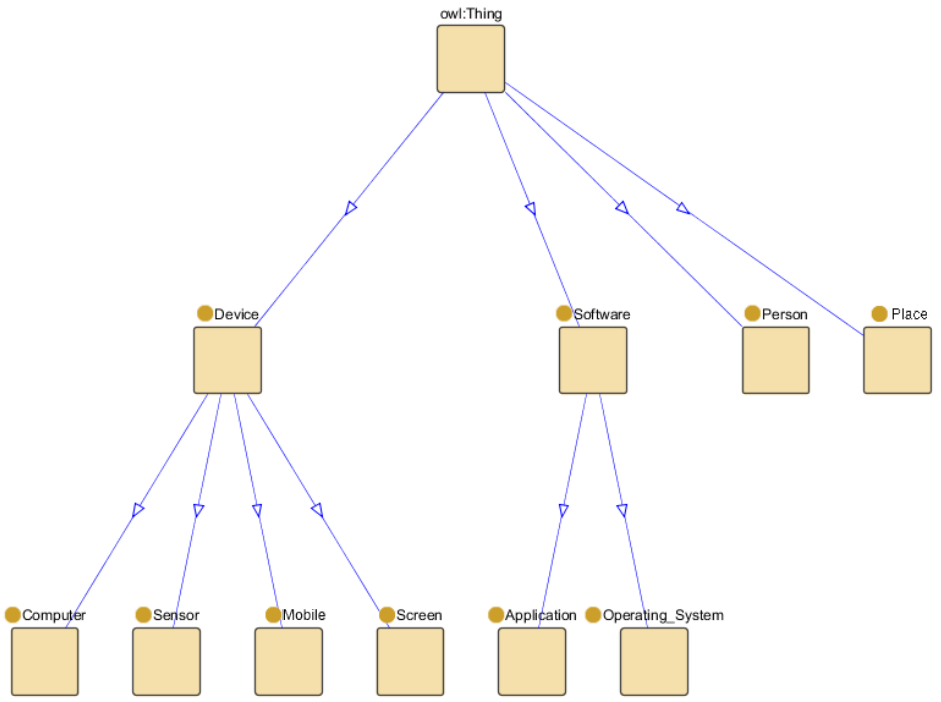

Figura 5.5: Hierarquia de classes para a ontologia de contexto da arquitetura proposta

5.4.2 Atributos

Os atributos ou propriedades de dado de uma classe são características que ela pode possuir e que serão usadas para diferenciá-las umas das outras, podendo também definir ações que podem ser executadas por essas classes (FREITAS, 2011).

As subclasses de uma classe abstrata herdam os atributos dessa classe abstrata. Isto é, se uma classe abstrata *Software* possuir um atributo Desenvolvedor, todas as classes que herdam dela (*Application* e *Operating System*) vão possuir esse mesma propriedade. As subclasses por sua vez podem possuir atributos únicos que não aparecem na sua classe-pai.

Os seguintes atributos são específicos da classe *Device* e de suas subclasses: *Processor, RAM, Resolution, Storage, BatteryPower, Frequency, Manufacturer*. A classe *Person*, que modela uma pessoa dentro do ambiente pervasivo, possui os seguintes atributos: *Name, Age, Height, isHandicapped*.

A classe Place tem como atributos *Height e Width*. Os elementos da classe Application possuem os atributos: *MinProc, MinRAM, MinRes, MinStorage, Developer*.

5.4.3 Relacionamentos

Os relacionamentos entre as classes da ontologia foram definidos baseados na necessidade de interação do sistema pervasivo em se adaptar as mudanças de contexto. Eles foram criados para possibilitar que as inferências sobre o contexto ocorram de forma simplificada, resultando em uma maior responsividade da ontologia.

Foram definidos 6 relacionamentos básicos: *connectedTo, contains, hasOperatingSystem, isAbleToRun, presentIn, runsOn*. A Tabela 5.2 auxilia na visualização desses relacionamentos.

Tabela 5.2: Relacionamentos presentes na ontologia da arquitetura proposta

Classe	Relacionamento	Classe	Relação Inversa	Simetrico
Computer	connectedTo	Screen	-	X
Screen	connectedTo	Computer	-	X
Place	contains	Device	presentIn	-
Place	contains	Person	presentIn	-
Device	hasOperatingSystem	Operating System	-	-
Device	isAbleToRun	Software	runsOn	-
Device	presentIn	Place	contains	-
Person	presentIn	Place	contains	-
Operating System	runsOn	Device	isAbleToRun	-

O relacionamento *connectedTo* é um relacionamento simétrico entre as classes *Computer* e

Screen, indicando que um computador está conectado a uma tela e *vice versa*. O relacionamento *contains* relaciona os *Devices* e *Persons* que estão contidos em uma *Place* e é completado pelo relacionamento reverso *presentIn*.

hasOperatingSystem é utilizado para indicar qual *Operating System* é executado em um *Device*. Esse relacionamento é inverso ao relacionamento *runsOn*. O relacionamento *isAbleToRun* liga um *Software* a um *Device* que pode executar esse software, de acordo com os atributos MinProc, MinRAM e MinRES.

É possível observar um grafo que apresenta o relacionamento entre as classes da ontologia na Figura 5.6.

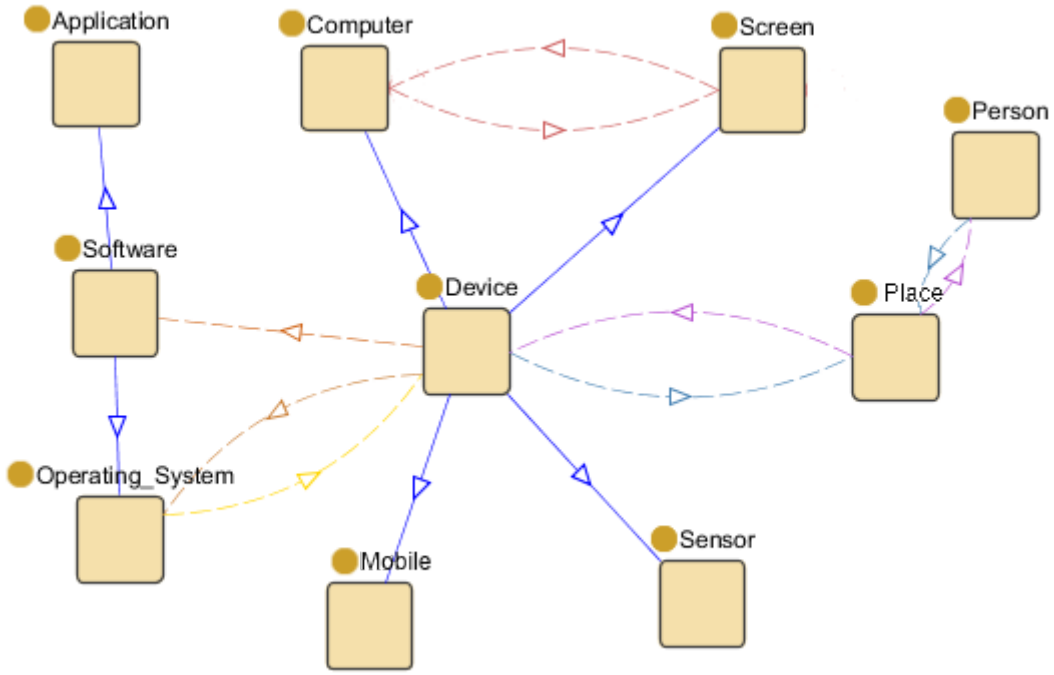

Figura 5.6: Grafo da ontologia de contexto

5.4.4 Instâncias

As instâncias de uma ontologia representam os indivíduos das diferentes classes definidas na ontologia que fazem parte do sistema. Cada objeto de alguma classe é considerado uma instância. Em um ambiente inteligente, as instâncias vão ser os elementos que farão parte do ambiente pervasivo: as pessoas serão instâncias da classe *Person*, os dispositivos computacionais serão instâncias de *Sensor* ou *Computer* ou *Mobile*, e assim por diante.

A ontologia de contexto da arquitetura proposta não tem nenhuma instância definida inicialmente, ela será populada de acordo com a utilização no sistema pervasivo.

5.5 Escalabilidade

Um dos principais diferenciais da arquitetura proposta é sua alta capacidade de acomodar novos dispositivos ou novos ambientes de computação pervasiva com a realização de um mínimo de esforço. A utilização da computação nas nuvens possibilita que o hardware utilizado para a computação pervasiva possa ser dimensionado de acordo com a demanda do ambiente. A arquitetura suporta dois modelos de escalabilidade: o horizontal (*scale out*) e o vertical (*scale up*).

5.5.1 Escalabilidade Horizontal

A escalabilidade horizontal nessa arquitetura consiste em adicionar mais servidores para atender as requisições do ambiente. A adição de novos servidores permite fazer com que novos ambientes pervasivos possam ser incluídos sem que os ambientes já existentes sejam afetados.

Caso seja necessário adicionar um quarto módulo local, por exemplo, deverá ser possível adicionar um novo Módulo de Ontologias e *Reasoning* para lidar com esse novo ambiente local sem a necessidade de interromper os outros ambientes que estão em execução. A Figura 5.7 exemplifica uma implementação da arquitetura com vários ambientes utilizando um mesmo Módulo de Monitoramento Remoto, porém Módulos de Ontologia e *Reasoning* diferentes.

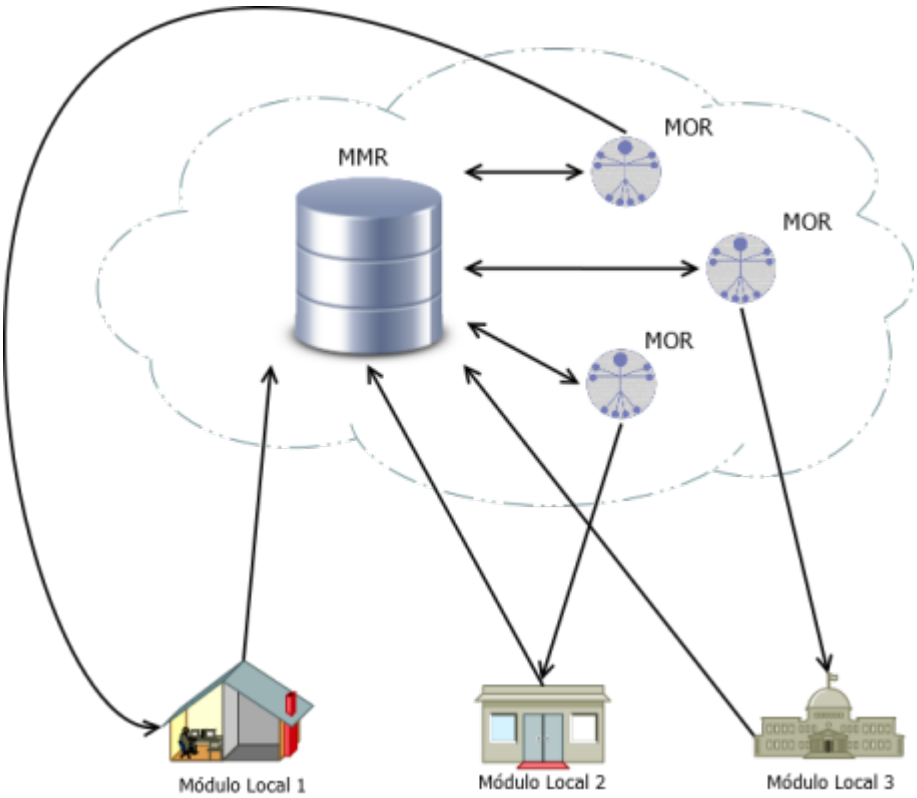

Figura 5.7: Arquitetura implementada com Escalabilidade Horizontal

5.5.2 Escalabilidade Vertical

A escalabilidade vertical nessa arquitetura consiste em aumentar a quantidade de recursos computacionais disponíveis para os componentes do Módulo Remoto, permitindo assim que o sistema seja capaz de processar mais requisições, sem a necessidade de adicionar mais máquinas para realizar o processamento. A utilização de um modelo de escalabilidade vertical permite também que os diferentes ambientes compartilhem informações de contexto pela utilização do mesmo Módulo de Monitoramento Remoto.

Por exemplo, para adicionar um novo módulo local, basta aumentar a quantidade de armazenamento disponível para o MML e aumentar o poder de processamento disponível para o MOR, não havendo necessidade de interferir com os ambientes pervasivos que já estão em execução. A Figura 5.8 mostra uma implementação da arquitetura contemplando vários ambientes utilizando um mesmo Módulo Remoto.

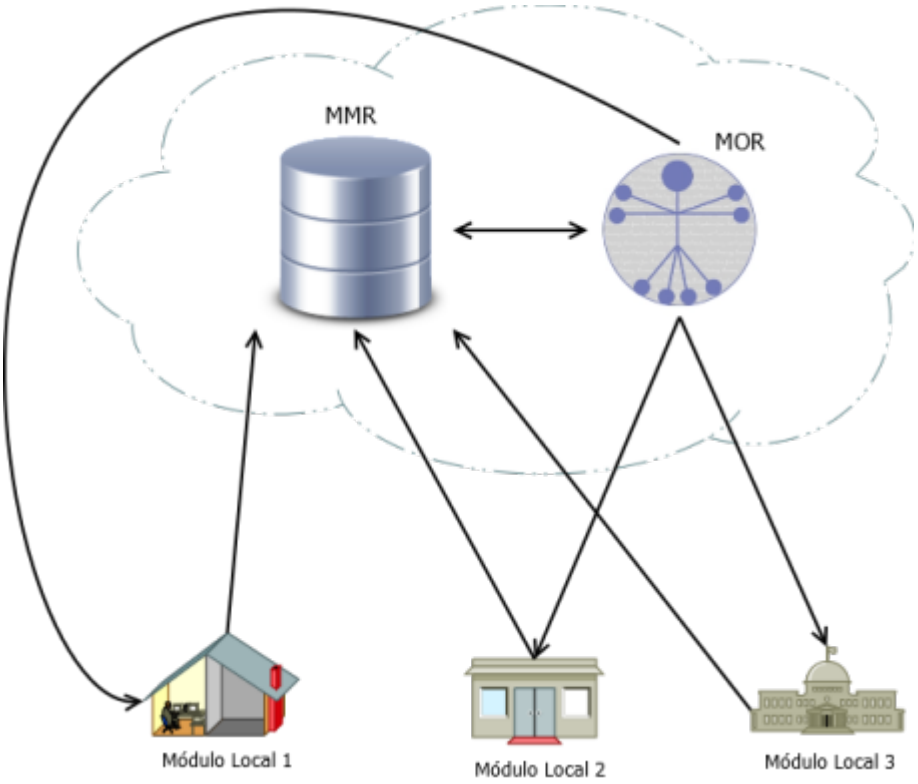

Figura 5.8: Arquitetura implementada com Escalabilidade Vertical

5.6 Sumário do Capítulo

Neste capítulo foi apresentada a proposta de arquitetura para utilização de *cloud computing* como maneira de auxiliar na computação pervasiva. Foram apresentados os requisitos, a descrição dos módulos, a ontologia para representação de contexto e os modelos de escalabilidade

que podem ser contemplados com a utilização de um sistema desenvolvido nessa arquitetura. Para permitir a realização de um comparativo entre a arquitetura proposta e as existentes, os trabalhos correlatos são apresentandos no Capítulo 7.

6 ESTUDO DE CASO

Com o propósito de verificar a aplicabilidade da arquitetura proposta, foi realizado um estudo de caso que será descrito ao longo deste capítulo. O estudo de caso buscou contemplar os aspectos diferenciados da arquitetura proposta e resultou em uma implementação funcional de um ambiente pervasivo.

6.1 Cenário

Optou-se por utilizar como cenário um ambiente de computação pervasiva residencial. A escolha desse cenário busca apresentar uma utilização prática da arquitetura proposta, tendo em vista que vários trabalhos da área de computação ubíqua e pervasiva utilizam ambientes residenciais para realização de testes.

O ambiente de testes (Módulo Local) foi composto por quatro computadores, um dispositivo móvel, quatro sensores de ruído, quatro sensores de luminosidade, quatro sensores de presença, software pervasivo sendo executado nos computadores e nos dispositivos móveis. Todos os componentes estão conectados através de uma rede de alta velocidade.

Em todos os cômodos foram instalados um computador, um sensor de ruído, um sensor de luminosidade e um sensor de presença. Cada um dos equipamentos possuí um identificador único, conforme modelado na ontologia de representação de contexto.

A aplicação pervasiva que gerencia esse ambiente é capaz de adaptar o conteúdo de acordo com a capacidade técnica do dispositivo computacional que está sendo utilizado. Sendo capaz de redimensionar o tamanho da fonte, aumentar ou diminuir o volume e controlar o nível de luminosidade dos *displays*. Em um dos computadores foi instalado um software capaz de retransmitir as informações de contexto para o Módulo Remoto.

O Módulo Remoto foi implementado utilizando o serviço de computação elástica da Amazon, o Amazon EC2. Foram utilizados dois nodos, um configurado com um banco de dados para armazenar as informações de contexto e outro dotado de um *reasoner* e responsável pela realização de inferências sobre o ambiente local. A Figura 6.1 apresenta uma visão geral da arquitetura implementada.

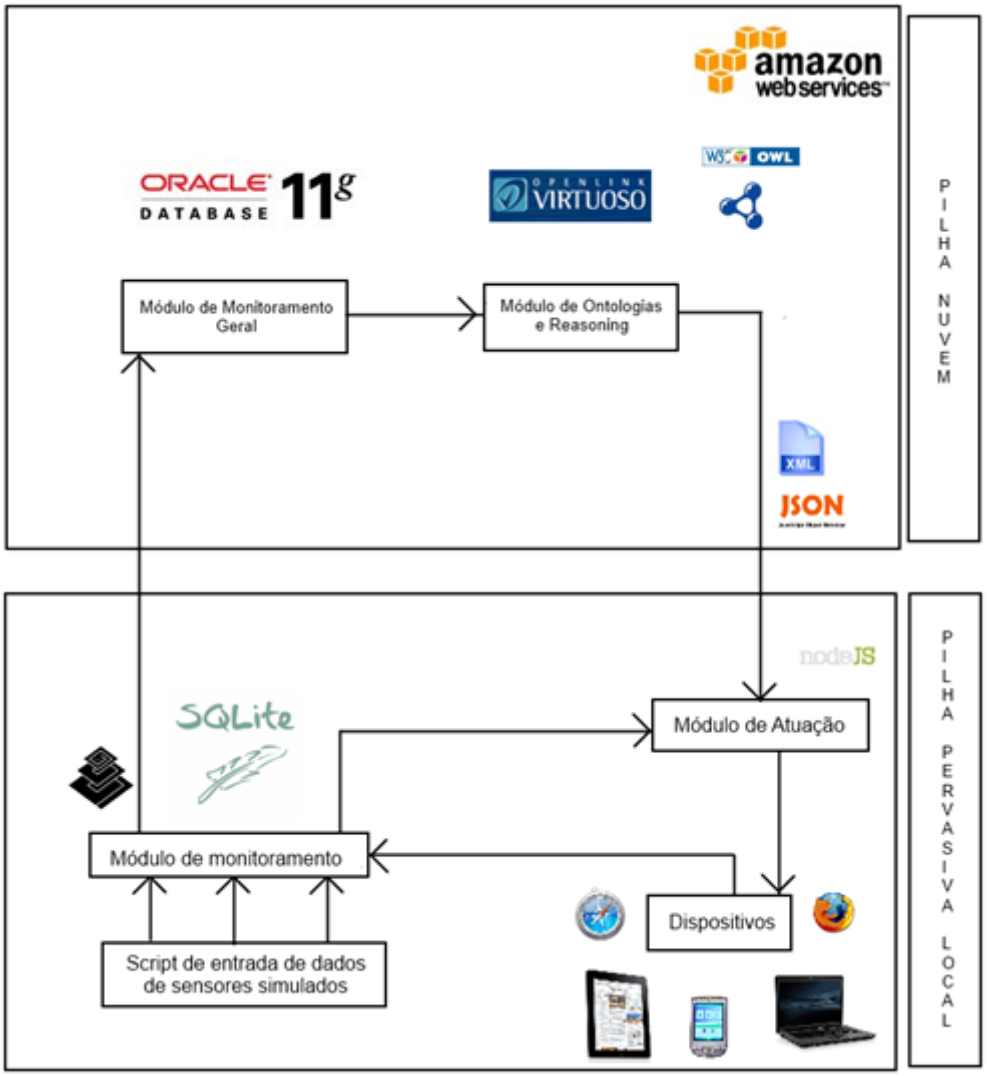

Figura 6.1: Visão geral da arquitetura implementada

6.2 Componentes Locais

O ambiente local foi implementado utilizando sensores de software. Os componentes locais são os dispositivos, sensores, módulos de atuação e módulos de monitoramento que realizam funções a nível do ambiente pervasivo. A Figura 6.2 apresenta uma visão dos componentes locais utilizados no estudo de caso.

6.2.1 Dispositivos

Para testar a capacidade da arquitetura proposta de trabalhar com dispositivos heterogêneos, cada um dos computadores foi configurado com características distintas: o Computador 1 (C1) executava o sistema operacional Windows 7 e possuia uma resolução de 1024x768 *pixels*; o Computador 2 (C2) operava com Windows 7 e uma resolução de 1920x1080 *pixels*; o Compu-

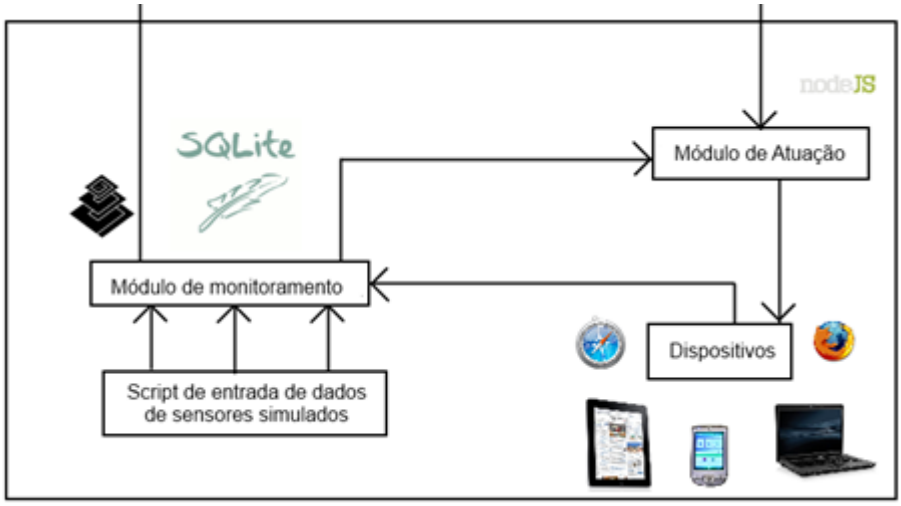

Figura 6.2: Visão dos Componentes Locais

tador 3 (C3) era um macbook com resolução de 1280x1024 *pixels* e OSX 10.6; o Computador 4 (C4) foi instalado com Ubuntu Linux 10.04 e uma resolução de tela de 1024x768 *pixels*.

Além desses dispositivos, um *tablet* (M1) com o sistema operacional Android e resolução de tela 1280x800 *pixels* foi utilizado. A Tabela 6.1 apresenta mais detalhes sobre a configuração dos dispositivos utilizados no estudo de caso.

Tabela 6.1: Demonstrativo entre os dispositivos utilizados no estudo de caso

	C1	**C2**	**C3**	**C4**	**M1**
S.O.	Windows 7	Windows 7	OSX 10.6	Ubuntu 10.04	Android 3.1
Resolução	1024x768	1920x1080	1280x1024	1024x768	1280x800
Processador	2GHz	6x 2.8GHz	2.2GHz	2GHz	2x 1.2GHz
Navegador	Chrome 16	Chrome 16	Chrome 16	Chromium 17	Opera Mobile

Todos os dispositivos possuiam um navegador capaz de realizar a exibição de conteúdo criado utilizando HTML5 e foram configurados para executar automaticamente o navegador apontado para um endereço específico assim que o computador fosse iniciado. Os dispositivos também encontravam-se conectados a uma rede de alta velocidade, os computadores através de conexão Ethernet e o dispositivo móvel utilizando uma rede sem-fio.

6.2.2 Sensores

Foram utilizados sensores simulados para representar sensores de ruído, sensores de luminosidade e sensores de presença. Cada um desses sensores foi pensado de maneira a representar fielmente um sensor real. Todos os sensores foram configurados para enviar informações diretamente ao Módulo de Monitoramento, através da rede Ethernet onde eles se encontravam

conectados.

O sensor de ruído tinha como utilidade detectar variações sonoras no cômodo onde fora instalado, realizando *pooling* aproximadamente uma vez a cada dez segundos. Os valores eram medidos em decibéis (dB). Variando de 0 até 127, onde 0 era o ruído padrão do ambiente vazio (silêncio absoluto).

Os sensores de luminosidade serviam para detectar o nível de luz presente no ambiente e fornecer essas informações para o Módulo de Monitoramento. Esses sensores enviavam a informação uma vez a cada cinco segundos. Eles utilizavam uma escala de 0 até 255 onde 0 é a escuridão total do ambiente e 255 é a iluminação total do ambiente.

O sensor de presença era capaz de detectar a presença de uma pessoa no cômodo. Ele foi projetado de forma assíncrona. Assim que uma pessoa era detectada no ambiente ele enviava um sinal para o Módulo de Monitoramento e assim que essa pessoa abandonava o ambiente ele enviava outro sinal para o Módulo de Monitoramento.

Foi desenvolvido um *script* em Python capaz de enviar as informações pertencentes aos sensores simuladores para o Módulo de Monitoramento. O *script* recebia as informações de controle de um arquivo de configuração e realizava a simulação dos sensores, enviando os dados através da rede.

6.2.3 Módulo de Monitoramento Local

O Módulo de Monitoramento Local foi desenvolvido utilizando o framework de rede Twisted em conjunto com um banco de dados relacional SQLite. O Twisted foi responsável pela implementação de sockets capazes de realizar a conexão com o *script* utilizado para simular os sensores e pelo envio da informação do ambiente local para a pilha remota.

Os dados relativos a representação do contexto eram armazenados em um banco de dados relacional, de forma sequêncial e enviados ao Módulo de Monitoramento Remoto utilizando uma conexão TCP. O Módulo de Monitoramento estava conectado ao Módulo de Monitoramento Remoto através de uma rede de alta velocidade e enviava as informações de contexto a cada 2 segundos. No ambiente local, o módulo de monitoramento era executado em *background* no computador C2.

6.2.4 Módulo de Atuação

O Módulo de Atuação foi desenvolvido utilizando node.js e estava presente na aplicação pervasiva sendo executada pelos navegadores do dispositivo móvel. Ele foi implementado uti-

lizando um servidor assíncrono para responder aos comandos enviados pelo Módulo Remoto de Ontologias e Reasoning. O Módulo de Atuação recebia os comandos enviados utilizando Javascript Serialized Object Notation (JSON) e executava esses comandos nos dispositivos do ambiente local.

O código responsável por receber os comandos do Módulo Remoto e executálos nos dispositivos pode ser observado em Código Fonte 6.1.

```
var io = require('node.io').listen();

io.sockets.on('connection', function (socket) {

  socket.on('message', function (msg) {

  switch(msg){
    case '+bright': require('brightness').more(10);
    break;
    case '-bright': require('brightness').less(10);
    break;
    case '+vol': require('volume').more(10);
    break;
    case '-vol': require('volume').less(10);
    break;
    case 'shut': require('fade').hide();
    break;
    case 'on': require('fade').show();
    break;
  }
});

  socket.on('disconnect', function () {
  socket.close();
  });
});
```

Código Fonte 6.1: Tratamento das Requisições de Mudança

6.3 Componentes Remotos

Os dois componentes remotos utilizados foram o Módulo de Monitoramento Remoto e o Módulo de Ontologias e Reasoning. Ambos os componentes foram implementados utilizando nodos presentes comercialmente no serviço de infraestrutura como serviço do Amazon EC2.

Os dois módulos foram implementados utilizando soluções *off the shelf* e padrões abertos para exemplificar a capacidade do arquitetura de se adequar as soluções já existentes no mercado. Cada um dos módulos foi executado em uma Instância Pequena do EC2, com um custo de aproximadamente $0,027 por hora. A Figura 6.3 apresenta uma visão dos componentes remotos utilizados no estudo de caso.

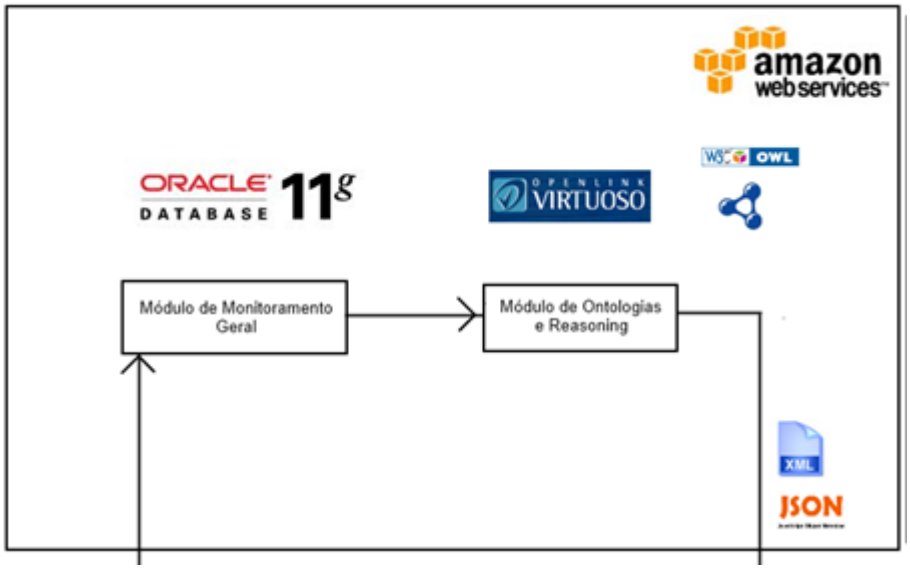

Figura 6.3: Visão dos Componentes Remotos

6.3.1 Ontologia Utilizada

A partir do modelo de ontologia fornecido pela arquitetura (Figura 5.5) foi criada uma nova ontologia extendendo as características especificas dos sensores presentes no estudo de caso. Essa nova ontologia reaproveitou as classes já existente e introduziu novas classes e relacionamentos.

Foram criadas três classes para representar esses sensores: *Brightness*, *Noise* e *Presence*. A classe *Brightness* foi criada para modelar os sensores de brilho, a classe *Noise* para modelar os sensores de ruído e a classe *Presence* para modelar os sensores de presença.

Também foi criado um relacionamento *sensedBy* entre *Presence* e *Person*. Esse relacionamento indica qual sensor do tipo *Presence* detectou qual pessoa presente no ambiente.

Para o desenvolvimento da ontologia foi utilizado a ferramenta Protégé e como linguagem de representação foi escolhida a OWL. A Figura 6.4 apresenta o grafo da ontologia utilizada no estudo de caso.

6.3.2 Consultas SPARQL

Foram definidas 6 consultas utilizando SPARQL para simular as funcionalidades do ambiente pervasivo: *volumeTooLow, volumeTooHigh, screenOff, screenOn, brightnessTooLow, brightnessTooHigh*. As consultas *volumeTooLow* e *volumeTooHigh* verificam se algum dos computadores que está em uso está com o som mais baixo do que o nível de ruído do ambiente e aumenta ou diminui o som conforme o resultado da consulta.

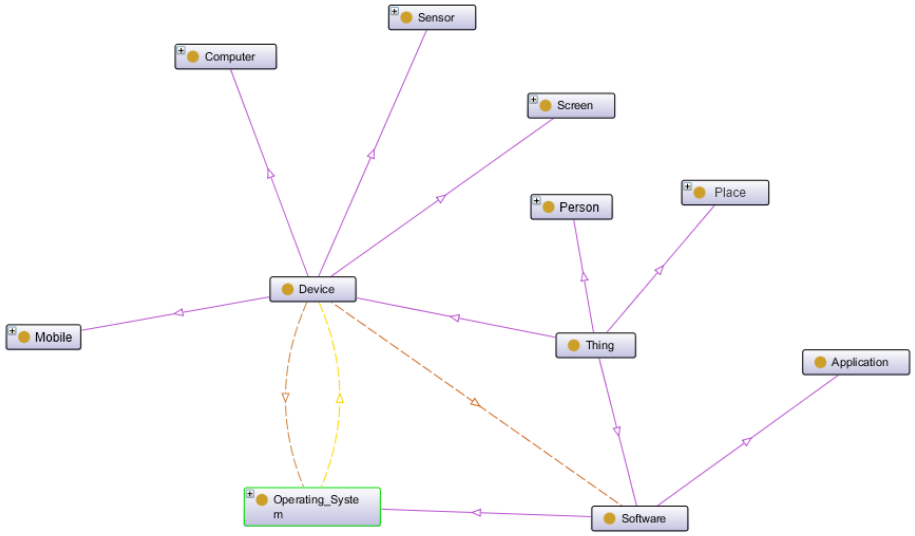

Figura 6.4: Ontologia utilizada no Caso de Uso

Já as consultas *screenOff* e *screenOn* habilitam ou desabilitam a tela do dispositivo de acordo com a presença de pessoas no cômodo. Caso nenhuma pessoa esteja presente a tela é desligada, sendo religada assim que uma pessoa é detectada no ambiente. O Código Fonte 6.2 mostra a consulta SPARQL responsável por verificar as telas que estão apagadas em ambientes com pessoas presentes.

As consultas *brightnessTooLow* e *brightnessTooHigh* verificam se a luminosidade da tela permite que o usuário interaja com o ambiente de forma agradável, caso isso não ocorra, os *displays* tem seu brilho ajustado.

```
1  PREFIX otl: <ambientelocal#>
2
3  SELECT ?screen ?room
4
5  WHERE {
6     ?room otl:contains otl:Person;
7     ?screen otl:presentIn ?room.
8  }
9
10 ORDER BY ?room
```

Código Fonte 6.2: Consulta screenOn

6.3.3 Módulo de Monitoramento Remoto

O Módulo de Monitoramento Remoto consistiu em uma máquina virtual executando Oracle Enterprise Linux Release 5 juntamente com um banco de dados Oracle Database 11g. O banco de dados estava configurado para aceitar conexões provenientes do ambiente local e do Módulo de Ontologias e Reasoning e armazenar os dados em uma tabela relacional.

Todas as atualizações no Módulo de Monitoramento Remoto resultavam em uma chamada para o Módulo de Ontologias e Reasoning que estava conectado utilizando uma rede de alta velocidade nos *datacenters* da Amazon.

6.3.4 Módulo de Ontologias e Reasoning

O Módulo de Ontologias e Reasoning foi implementado utilizando o software Virtuoso, desenvolvido pela OpenLink. O Módulo de Ontologias e Reasoning era responsável pelo armazenamento do modelo ontológico representativo do contexto do ambiente local e pela realização das inferências em cima dessa ontologia.

Como a informação foi armazenada utilizando bancos de dados relacionais tanto no Módulo de Monitoramento Local quanto no Módulo de Monitoramento Remoto, foi necessário a criação de um mapeamento para popular a ontologia.

O processo de mapeamento consiste em 4 etapas:

* Escolha da base de dados relacional
* Escolha da ontologia para exposição dos dados
* Identificação das tabelas que armazenam a informação de contexto
* Criação de relacionamentos um para um realizando a instanciação dos dados.

A Figura 6.5 apresenta o processo de mapeamento realizado no Virtuoso para permitir a integração entre o Módulo de Monitoramento Remoto e o Módulo de Ontologias e Reasoning.

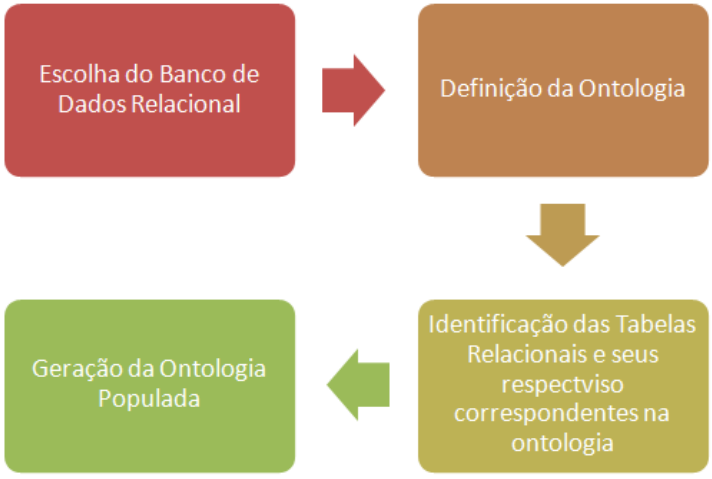

Figura 6.5: Processo de Mapeamento

A Figura 6.6 apresenta a Ontologia populada com o caso de uso e permite uma visão dos relacionamentos definidos pelo modelo ontológico. O processo de mapeamento dos dados relacionais para instâncias da ontologia é importante, pois permite que as inferências possam ser realizadas utilizando SPARQL.

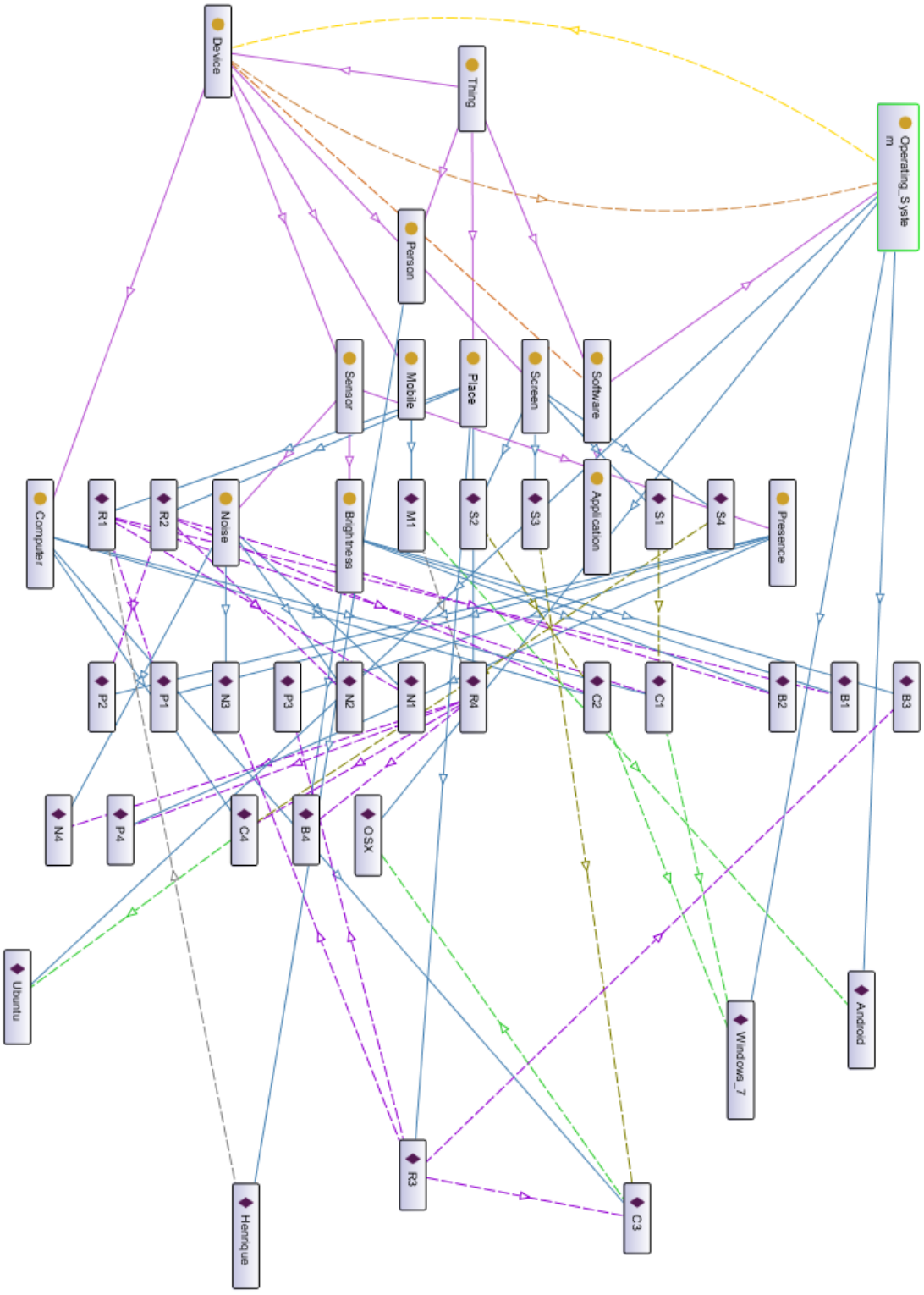

Figura 6.6: Ontologia Populada a partir do processo de mapeamento relacional-ontológico

6.4 Resultados

Os testes de *performance* foram realizados no Módulo de Ontologias e Reasoning conforme o Módulo de Monitoramento Remoto capturava as informações vindas do Módulo de Monitoramento Local. Foram executados 10 testes nas instâncias Pequena, Média e Extragrande do Amazon EC2. A Tabela 6.2 apresenta um comparativo entre os recursos disponíveis em cada uma das instâncias, onde cada unidade computacional EC2 corresponde a aproximadamente a um processador de 1.7GHz.

Tabela 6.2: Instâncias utilizadas nos testes

Instância	**Memória**	**Armazenamento**	**Poder Computacional**
Pequena	1,7GB	160GB	1 núcleo com 1 EC2
Média	3,75GB	410GB	2 núcleos com 1 EC2
Extragrande	15GB	1.690GB	4 núcleos com 2 EC2

A Figura 6.7 apresenta o comparativo em relação ao número médio de consultas executadas por segundo de acordo com o aumento na quantidade de instâncias presentes na ontologia. É possível identificar que todas as instâncias apresentam uma queda no número de consultas conforme a quantidade de instâncias na ontologia aumenta, o número de consultas executadas por segundo tende a diminuir.

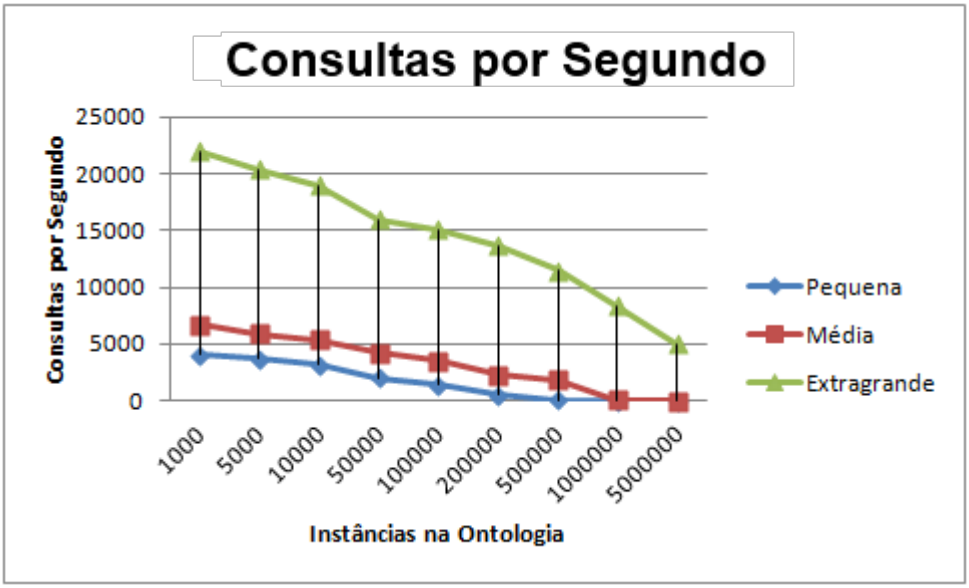

Figura 6.7: Número Médio de Consultas Executadas por Segundo

Com um número pequeno de instâncias na ontologia, as operações de inferência são realizadas de forma mais rápida, mas a medida que o número dessas instâncias aumenta, o poder computacional necessário para realizar as consultas aumenta e com isso o desempenho apresenta

uma queda. Mesmo com 5 milhões de instâncias a máquina Extragrande do EC2 conseguiu um elevado número de consultas por segundo, muito além do que é previsível em uma situação real.

6.5 Sumário do Capítulo

Esse capítulo apresentou um estudo de caso controlado, em um ambiente residencial simulado, utilizando a arquitetura proposta no trabalho. Tanto o Módulo Local quanto o Módulo Remoto foram implementados seguindo as especificações e os requisitos propostos pelo modelo.

A ontologia sugerida foi extendida, dando origem a uma ontologia que foi capaz de atender as consultas realizadas no estudo de caso. Os componentes do Módulo Remoto foram implementados utilizando tecnologias disponíveis no mercado e executados na nuvem computacional da Amazon.

7 TRABALHOS RELACIONADOS

Para permitir uma melhor análise da arquitetura proposta serão apresentadas arquiteturas correlatas para ambientes de computação pervasiva. Ao longo dos anos, várias arquiteturas e *middlewares* para sistemas pervasivos foram propostos.

Optou-se por escolher trabalhos que permitissem a criação de ambientes inteligentes utilizando consciência de contexto. Foram escolhidas as arquiteturas MIDAS (MEDVIDOVIC; MALEK, 2007), ISAMpe (AUGUSTIN et al., 2002b), OntoHealth (GASSEN, 2010) e CoBrA (CHEN; FININ; JOSHI, 2003).

7.1 CoBrA

O *Context Broker Architecture*, ou CoBrA é uma arquitetura baseada em agentes para auxiliar na computação consciente de contexto em ambientes inteligentes (YE et al., 2007). A arquitetura foi desenvolvida levando em consideração três pontos principais: a necessidade de uma ontologia comum, um modelo compartilhado de contexto e uma política de controle de quais informações de contexto os usuários gostariam de compartilhar.

Ela explora a utilização de linguagens da Web Semântica para definir uma ontologia de contexto e compartilhar informações sobre contexto e inferências em cima dessa informação (CHEN; FININ; JOSHI, 2003). A informação de contexto, nessa arquitetura, é representada através de um conjunto de ontologias chamado de CoBrA-ONT, implementado utilizando OWL (CHEN; FININ; JOSHI, 2004). Nessas ontologias ficam definidos conceitos e relações descrevendo locais físicos, pessoas, agentes de software, dispositivos móveis e eventos.

Para permitir o gerenciamento da informação de contexto e permitir o compartilhamento dessas informações, a arquitetura utiliza um *context broker* (corretor ou agente de contexto, em tradução livre). O agente de contexto pode inferir informações sobre o contexto que não poderiam ser detectadas pelos sensores físicos e também é capaz de resolver inconsistências que podem decorrer de informações imprecisas vindas dos sensores. A Figura 7.1 apresenta um exemplo de detecção de incoerências utilizando o *context broker*.

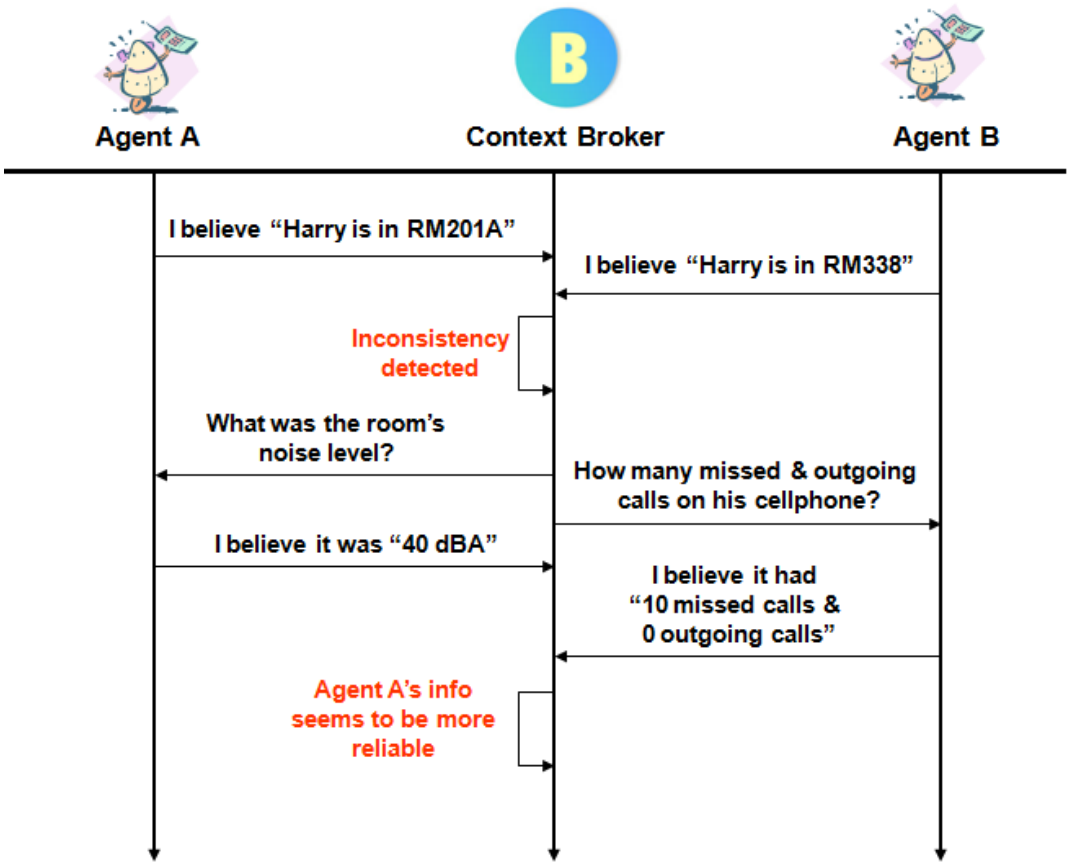

Figura 7.1: Resolução de conflitos usando o *context broker* (CHEN; FININ; JOSHI, 2004)

Nesse exemplo dois agentes acreditam saber a localização do usuário Harry. O Agente A é implementado utilizando um software de conhecimento de voz, enquanto o Agente B usa um sensor Bluetooth no celular. O Agente A acredita que Harry se encontra na sala RM201A enquanto o Agente B detecta que Harry está em RM338. O agente de contexto detecta a inconsistência e envia duas mensagens, uma para cada agente, buscando resolver essa incoerência.

A mensagem enviada para o Agente A pergunta qual o nível de ruído na sala RM201A e a enviada para o Agente B pergunta quantas chamadas foram realizadas pelo celular e quantas ligações foram perdidas. O Agente A responde indicando que o nível de ruído é de 40dBA na sala RM201A e a resposta do Agente B é de que nenhuma chamada foi realizada e 10 ligações foram perdidades pelo celular que se encontra em RM338. A partir dessas informações de contexto, o *context broker* entende que Harry está na sala RM201A e que ele esqueceu seu celular em RM338.

O ambiente inteligente implementado utilizando CoBrA pode conter informação de contexto do ambiente vindo de sensores inteligentes (*tags* RFID), sensores ambientais ou dos próprios dispositivos, e de informação. A informação de contexto também pode ser obtida de

fontes externas como servidores de calendário, de e-mail, bancos de dados ou entidades que não pertencem ao ambiente. A Figura 7.2 apresenta uma visão geral da arquitetura.

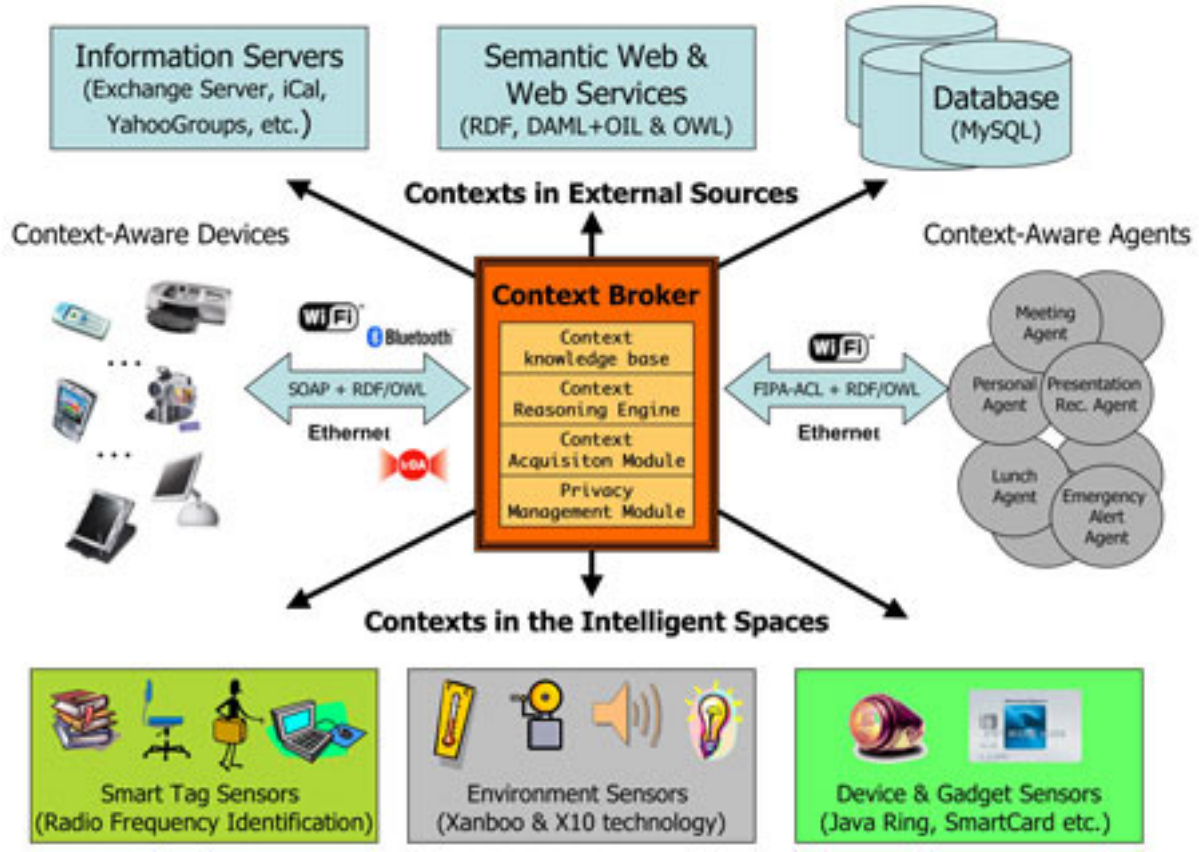

Figura 7.2: Visão geral da arquitetura Cobra (CHEN; FININ; JOSHI, 2003)

A CoBrA foi uma das primeiras arquiteturas a possuir uma entidade central capaz de gerenciar e manter a informação de contexto (YE et al., 2007). Todos as entidades computacionais do ambiente possuem conhecimento prévio do *context broker* e os agentes de alto nível se comunicam com ele utilizando uma linguagem de comunicação de agentes (CHEN; FININ; JOSHI, 2004). Outras características interessantes dessa arquitetura são a capacidade de realizar inferências baseadas em informações temporais e espaciais, ambas utilizando *reasoners* criados especificamente para atender essa arquitetura.

7.2 MIDAS

A arquitetura MIDAS é uma arquitetura simples para a computação pervasiva com base na utilização de sensores (MEDVIDOVIC; MALEK, 2007). Os ambientes computacionais são estáticos e os dispositivos computacionais são em sua maioria PDAs conectados a um computador central que realiza todas as computações sobre o ambiente pervasivo.

Ele é desenvolvida sobre uma família de sensores chamados MIDAS. Esses sensores se comunicam através de uma conexão sem fio com uma série de *gateways* conectados a um *hub* central que é o núcleo do ambiente pervasivo. Esse concentrador central é responsável pelo compartilhamento de informações de contexto com as aplicações pervasivas e os dispositivos computacionais presentes no ambiente. A figura 7.3 exibe um modelo geral da arquitetura.

Figura 7.3: Arquitetura pervasiva MIDAS (MEDVIDOVIC; MALEK, 2007)

Os sensores são utilizados para monitorar o ambiente e comunicar seu status entre si e para os *gateways*. Os *gateways* são responsáveis pelo gerenciamento dos sensores, decodificando, agregando e fundindo os dados recebidos dos sensores antes de propagar esses dados para o concentrador.

O concentrador reune as informações de todos os sensores e permite que o usuário do sistema interaja e modifique tanto os sensores quanto os *gateways* de uma interface de comando. Ele também propaga os dados dos sensores para os PDAs que estão sendo utilizados dentro do sistema (MEDVIDOVIC; MALEK, 2007).

Para realizar a implementação do sistema MIDAS foi desenvolvida uma Máquina Virtual Modular (MVM), projetada em C++ como uma camada de abstração em cima de vários sistemas operacionais (Linux, Windos, eCos) e plataformas de hardware (x86, KwikByte e outras plataformas de sensores proprietárias) (MALEK et al., 2007). Isso foi feito para permitir a flexibilidade da arquitetura para suportar novos sistemas operacionais ou dispositivos de hardware, necessitando apenas desenvolver versões específicas da MVM para o dispositivo a ser inserido no sistema.

Como o principal foco dessa arquitetura são dispositivos móveis, o MIDAS possui um sistema de detecção de falhas implementado por software. Ele é capaz de detectar problemas como a desconexão da rede, o desaparecimento dos dispositivos do sistema e problemas com a leitura dos sensores. Tudo isso é implementado através de classes em C++ (MEDVIDOVIC; MALEK, 2007).

7.3 ISAMpe

O ambiente pervasivo do projeto ISAM (Infraestrutura de Suporte às Aplicações Móveis) permite que aplicações em um ambiente pervasivo sejam móveis, flexíveis e adaptáveis (AUGUSTIN et al., 2002b). O ambiente ISAM contempla mais do que apenas um ambiente pervasivo, é um arcabouço de desenvolvimento de software, com metodologias e práticas próprias (AUGUSTIN et al., 2002a).

O ambiente pervasivo é formado por células (EXEHDAcells), que são compostas por nós fixos (EXEHDAnodes) conectados a uma rede cabeada e nós móveis (EXEHDAmob-nodes) conectados a rede sem fio. Toda célula pervasiva possui um computador central (EXEHDAbase) com a função de identificação, autenticação e liberação das funcionalidades do sistema. As várias EXEHDAcells trocam informações de contexto entre si. A estrutura de componentes desse ambiente pervasivo pode ser observada na Figura 7.4.

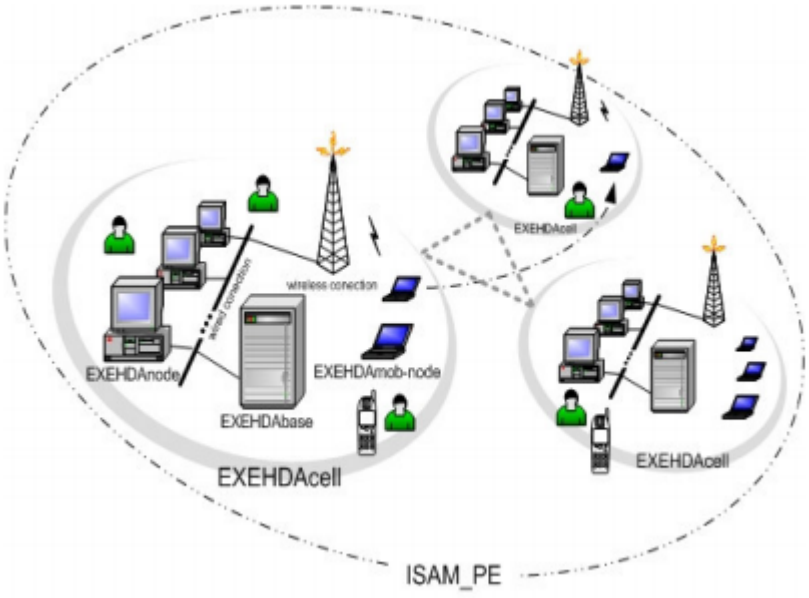

Figura 7.4: Componentes do ISAMpe (AUGUSTIN et al., 2002a)

7.3.1 Middleware EXEHDA

O EXEHDA (YAMIN, 2004) é um *middleware* para computação adaptativa ao contexto que compõe o projeto ISAM. Ele é baseado em serviços e que tem como objetivo criar e gerenciar ambientes ubíquos, promovendo a execução de aplicações com semântica *follow-me* (LOPES; PILLA; YAMIN, 2007).

Os principais requisitos do EXEHDA são a capacidade de arcar com a dinamicidade e heterogeneidade dos ambientes de processamento onde a arquitetura pervasiva é implementada e o suporte a semântica *follow-me*, levando em consideração as diferenças de *hardware* presentes nos dispositivos computacionais e as diferentes infraestruturas para conexão as redes de computadores.

O EXEHDA permite a movimentação entre dispositivos de artefatos de software e as informações de contexto relevantes, levando em consideração o deslocamento do usuário. Isso permite que uma mesma aplicação possa transitar entre diferentes dispositivos com a mesma informação de contexto.

7.4 OntoHealth

A arquitetura proposta por GASSEN (2010) contempla ambientes hospitalares de computação pervasiva. Ela foi desenvolvida tendo em vista que o ambiente hospitalar é diferente de uma universidade ou de uma casa, pois possui entidades "limitadas"(GASSEN, 2010).

As informações de um paciente armazenadas em um Prontuário Eletrônico do Paciente (PEP), por exemplo, só podem ser compartilhadas com entidades específicas (médicos, enfermeiros e pessoas autorizadas pelo paciente a visualizar essas informações). A Figura 7.5 mostra a visão da arquitetura OntoHealth.

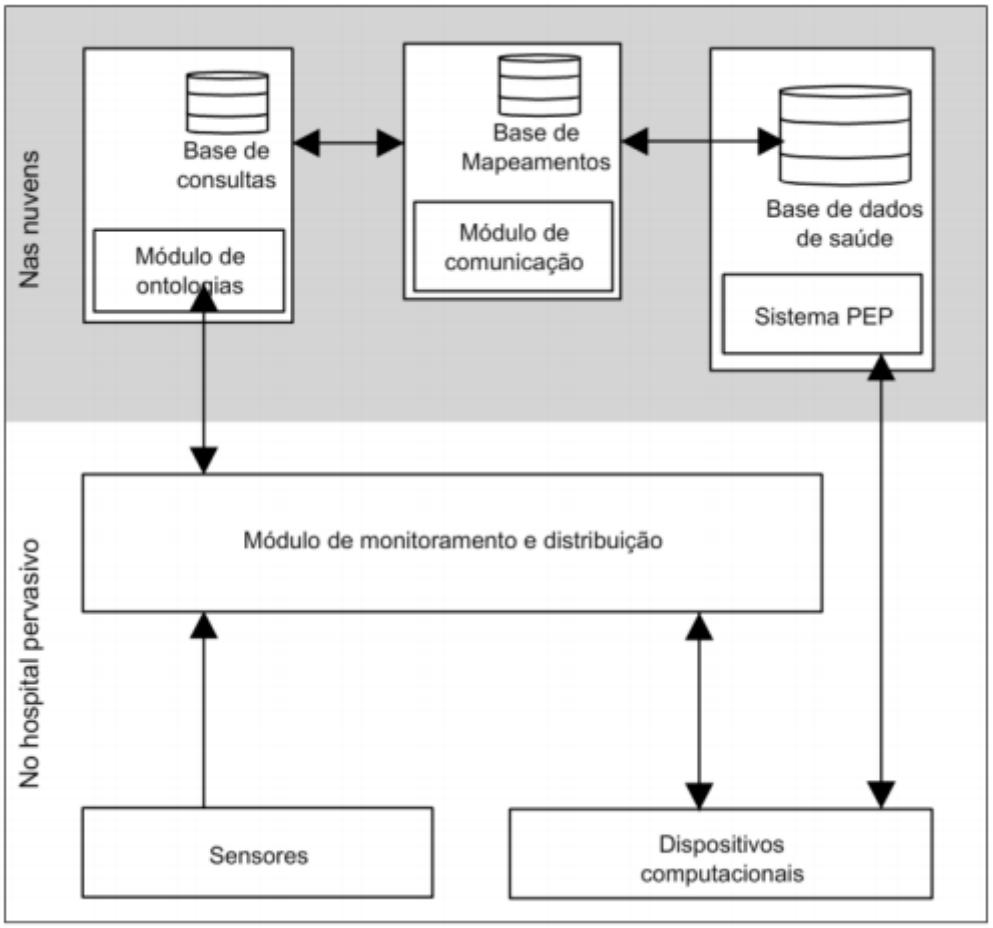

Figura 7.5: Arquitetura do Sistema (GASSEN, 2010)

A informação de contexto nesse sistema é gerenciada com o uso de ontologias, onde cada ontologia descreve um ambiente específico do hospital, reduzindo assim o número de entidades descritas em cada ambiente e aumentando o desempenho do processamento e a realização de inferências sobre o ambiente. Na arquitetura OntoHealth, essas ontologias são descritas utilizando OWL. As ontologias também fazem o uso de regras SWRL e as consultas são feitas através de SQWRL ou SPARQL.

É possível observar que o ambiente pervasivo é composto por sensores, dispositivos computacionais e um Módulo de Monitoramento e Distribuição. Esse módulo de monitoramento e distribuição é responsável por centraliza a informação do ambiente, detectando a informação enviada pelos dispositivos e eventos de modificação de contexto. Os dados armazenados por esse módulo ficam em um documento XML que será enviado para a ontologia do ambiente posteriormente (GASSEN, 2010).

Na nuvem é possível encontrar o Módulo do Sistema PEP, com a função de armazenar as informações sobre os pacientes e as aplicações disponíveis para o ambiente, o Módulo de

Ontologias, onde são realizadas as inferências sobre o ambiente e a política de controle de acesso a informações, e um Módulo de Comunicação para interligar os dois módulos anteriores.

Dentre os diferenciais desse sistema podem ser destacadas a estensibilidade do sistema e sua capacidade de interoperabilidade e personalização através da utilização de ontologias para modelar o ambiente.

7.5 Comparativo entre as arquiteturas pervasivas

Ao realizar uma análise das características presentes na arquitetura proposta é possível delinear uma comparação com as arquiteturas de computação ubíqua e pervasiva existentes. Foram analisados alguns quesitos: especificidade da arquitetura computacional, mobilidade de informação, mobilidade de aplicações, realização de inferências sobre o contexto e modelo de arquitetura.

O quesito *especificidade da arquitetura* leva em consideração a capacidade da arquitetura de se adaptar a diferentes cenários sem a necessidade de alterações na sua estrutura lógica. *Mobilidade de Informação* leva em consideração a capacidade de compartilhar informações de contexto ao longo do ambiente pervasivo. *Mobilidade de Aplicação* é a capacidade de mover aplicações em execução para outros dispositivos pervasivos de maneira proativa e sem perdas. A Tabela 7.1 mostra esse comparativo.

Tabela 7.1: Comparativo entre as arquiteturas pervasivas

Arquitetura	**Especificidade**	**Mob. de Informação**	**Mob. de Aplicações**
CoBrA	Genérica	Global	Global
OntoHealth	Específica	Global	Local
ISAMpe	Genérica/Adaptável	Global	Global
Midas	Genérica/Adaptável	Local	Local
Proposta	Genérica/Adaptável	Global	Local

O nível de especifidade pode ser classificado em Genérica, Adaptável ou Específica. A classificação Genérica indica que a arquitetura disponibiliza um modelo básico que pode ser utilizado por qualquer ambiente pervasivo, sem a necessidade de alterações. Adaptável permite modificar o modelo genérico para atender situações específicas. E a classificação Específica indica que o modelo da arquitetura foi definido para um fim específico e alterações nesse modelo implicariam na perda da essência do ambiente.

Tanto a arquitetura CoBrA quanto a ISAMpe contemplam a mobilidade de aplicações ao longo do ambiente pervasivo, desde que esses dispositivos possam executar sistemas operacio-

nais e aplicações compatíveis e especificamente desenvolvidas para o ambiente. A arquitetura proposta na dissertação não aborda esse conceito, limitando a mobilidade das aplicações a telas em um mesmo ambiente mas permitindo um maior número de dispositivos heterogêneos no ambiente.

A mobilidade de informação é garantida pela utilização do Módulo de Monitoramento Remoto, responsável por armazenar dados de contexto de todos os ambientes locais na arquitetura proposta. Isso permite que módulos remotos tenham acesso a informações do contexto de outros ambientes sem nenhum problema.

Outros dois aspectos importantes são o *modelo de inferências sobre o contexto* utilizado pela arquitetura pervasiva e o *modelo de arquitetura computacional* em que ela foi desenvolvida. Esse comparativo é realizado na Tabela 7.2.

Tabela 7.2: Comparativo entre as arquiteturas pervasivas - parte 2

Arquitetura	**Inferências sobre contexto**	**Arquitetura Computacional**
CoBrA	Baseado em Ontologias	Agentes Distribuídos
OntoHealth	Baseado em Ontologias	Cloud
ISAMpe	Misto	Grid
Midas	Baseado em Regras de Código	PC
Proposta	Baseado em Ontologias	Cloud

A tendência na utilização de *reasoning* baseado em ontologias fica clara ao comparar a Tabela 7.2. O único sistema pervasivo a não utilizar um sistema de inferências baseado em Ontologias é o Midas (MEDVIDOVIC; MALEK, 2007). A flexibilidade alcançada para a representação do contexto é muito importante em sistemas pervasivos. Os resultados alcançados na utilização de ontologias no trabalho são similares aos encontrados por AY (2008).

7.6 Sumário do Capítulo

Embora existam diversos modelos de arquiteturas para a computação pervasiva, cada uma delas contempla cenários específicos. É possível notar algumas tendências no desenvolvimento dessas arquiteturas.

As arquiteturas tendem a centralizar a informação de contexto em um dispositivo. Seja ele um agente de software (CoBrA), um módulo de banco de dados (OntoHealth) ou um concentrador de hardware (MIDAS).

Todas as arquiteturas tentam permitir a adição de novos dispositivos ao ambiente de computação pervasiva com o menor esforço possível. A utilização de ontologias tem permitido que

os ambientes se tornem mais heterogêneos e flexíveis, removendo a necessidade de que novos dispositivos sejam adicionados possuindo hardware e software específicos para funcionar no ambiente. O comparativo entre a arquitetura proposta e as arquiteturas relacionadas permite uma melhor visualização das características e diferenciais da arquitetura proposta.

8 CONCLUSÃO

Nos dias de hoje, computadores já fazem parte do nosso dia-a-dia: possuímos telefones pequenos e rápidos, *tablets* com poder de processamento maior do que *mainframes* dos anos 1990, notebooks e computadores pessoais já não são mais artigos de luxo e podem ser encontrados sem dificuldade tanto em casa quanto no trabalho. Essa popularização dos computadores e dos dispositivos móveis faz com a visão de computação pervasiva como a proposta por WEISER (1991) venha se tornando uma realidade.

Porém, ainda existem algumas dificuldades que precisam ser enfrentadas para facilitar a criação de ambientes pervasivos. Mesmo com o avanço nesta área, as arquiteturas existentes fazem uso de computadores locais com alto poder de processamento como centro das computações para o funcionamento do ambiente pervasivo, o que acaba por limitar a capacidade operacional desses ambientes.

Este trabalho foi motivado pela falta de arquiteturas para o desenvolvimento de ambientes de computação pervasiva utilizando recursos disponíveis em uma nuvem computacional. A nuvem computacional poderia resolver problemas como o aumento crescente no número de dispositivos presentes em um ambiente pervasivo sem que fosse necessário interromper o ambiente pervasivo para realizar alterações de *hardware*, permitindo também uma redução dos custos associados a criação desses ambientes.

Apresentou-se uma descrição detalhada de uma arquitetura que possa servir de base para a criação de ambientes pervasivos utilizando conceitos de computação pervasiva, ontologias e computação em nuvem. A arquitetura foi dividida em dois módulos (um módulo local e um módulo remoto).

O Módulo Local realiza o sensoreamento das informações de contexto e apresenta os dispositivos presentes fisicamente no ambiente pervasivo, enquanto o Módulo Remoto é responsável pela realização das inferências acerca da informação recebida pelo módulo local. O Módulo Remoto é implantado na nuvem computacional, permitindo que os recursos computacionais sejam alocados sob demanda.

Para que essas inferências pudessem ser realizadas, foi desenvolvida uma ontologia capaz de representar as informações básicas de contexto do ambiente. A ontologia proposta pode ser reutilizada ou estendida na criação de outros ambientes pervasivos.

A arquitetura foi validada utilizando um estudo de caso controlado. A simulação de um

ambiente residencial mostrou a viabilidade da utilização do modelo proposto, utilizando tecnologia já existente, envolvendo tanto bancos de dados relacionais quanto *reasoners* comerciais. Estendeu-se a ontologia para representação de contexto sugerida pela arquitetura de modo a acomodar os sensores existentes no ambiente.

A utilização de ontologias para a representação de infomação do contexto foi um fator muito positivo na utilização da arquitetura proposta, permitindo o reuso de informação e a integração simples entre sistemas de inferência e bancos de dados tradicionais.

Levando em considerações os requisitos elencados para que a arquitetura fosse desenvolvida (baixo consumo de recursos no ambiente local, performance e escalabilidade, habilidade de suportar uma vasta gama de dispositivos e sistemas operacionais e o uso de ontologias), acredita-se que a utilização da arquitetura proposta pode vir a facilitar a criação e manutenção de ambientes pervasivos, reduzindo os custos operacionais e possibilitando um crescimento no número de dispositivos presentes no ambiente de forma transparente e natural.

Os resultados alcançados durante a realização do trabalho contemplam os objetivos almejados no seu começo. Tanto a computação em nuvem quanto a computação pervasiva ainda tem muito a evoluir, mas é opinião do autor que as duas áreas podem se beneficiar da arquitetura proposta.

8.1 Trabalhos Futuros

A criação de sistemas com a arquitetura proposta tendem a facilitar a criação de ambientes de computação pervasiva e acarretar em uma redução de custo e trabalho. Porém muito ainda pode ser feito.

Um estudo compreensivo sobre os custos da implementação da arquitetura definida nesse trabalho. Juntamente com uma comparação com os custos atuais da criação de um ambiente pervasivo podem acrescentar bastante a área do conhecimento e adicionar valor.

A ontologia genérica de contexto pode ser adaptada para diferentes cenários, podendo ser utilizada não só em um ambiente residencial pervasivo, mas em ambientes hospitalares, escritórios pervasivos, ambientes de ensino e aprendizagem.

A implantação da arquitetura proposta para servir como base em sistemas de hospitais pervasivos pode auxiliar que trabalhos como GASSEN (2010); FREITAS (2011) se tornem uma realidade cada vez mais presente no nosso dia-a-dia.

REFERÊNCIAS

AGRAWAL, D.; DAS, S.; EL ABBADI, A. Big data and cloud computing: current state and future opportunities. In: INTERNATIONAL CONFERENCE ON EXTENDING DATABASE TECHNOLOGY, 14., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.530–533. (EDBT/ICDT '11).

AKHANI, J.; CHUADHARY, S.; SOMANI, G. Negotiation for resource allocation in IaaS cloud. In: FOURTH ANNUAL ACM BANGALORE CONFERENCE, 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.15:1–15:7. (COMPUTE '11).

APPAVOO, J.; WATERLAND, A.; DA SILVA, D.; UHLIG, V.; ROSENBURG, B.; VAN HENSBERGEN, E.; STOESS, J.; WISNIEWSKI, R.; STEINBERG, U. Providing a cloud network infrastructure on a supercomputer. In: ACM INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 19., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.385–394. (HPDC '10).

AUGUSTIN, I.; YAMIN, A. C.; BARBOSA, J. L. V.; GEYER, C. F. R. ISAM, a software architecture for adaptive and distributed mobile applications. In: ISCC, 2002. **Anais...** IEEE Computer Society, 2002. p.333–338.

AUGUSTIN, I.; YAMIN, A. C.; BARBOSA, J. L. V.; GEYER, C. F. R. ISAM, a Software Architecture for Adaptive and Distributed Mobile Applications. In: SEVENTH INTERNATIONAL SYMPOSIUM ON COMPUTERS AND COMMUNICATIONS (ISCC'02), 2002, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2002. p.333–. (ISCC '02).

AY, F. Context Modeling and Reasoning using Ontologies. In: IEEE COMPUTER SOCIETY, 2008. **Anais...** [S.l.: s.n.], 2008.

BARDRAM, J. E. Applications of context-aware computing in hospital work: examples and design principles. In: ACM SYMPOSIUM ON APPLIED COMPUTING, 2004., 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.1574–1579. (SAC '04).

BHASKAR, P.; RIMAL, E. C.; LUMB, I. A Taxonomy, Survery, and Issues of Cloud Computing Ecosystems. In: **Cloud Computing**: principles, systems and applications. [S.l.]: Springer, 2010. (Computer Communications and Networks).

BJORK, S.; HOLOPAINEN, J.; LJUNGSTRAND, P.; AAKESSON, K.-P. Designing Ubiquitous Computing Games. **Personal Ubiquitous Comput.**, London, UK, v.6, p.443–458, January 2002.

BOCK, C.; FOKOUE, A.; HAASE, P.; HOEKSTRA, R.; HORROCKS, I.; RUTTENBERG, A.; SATTLER, U.; SMITH, M. **OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax**. 2009.

BOZSAK, E.; EHRIG, M.; HANDSCHUH, S.; HOTHO, A.; MAEDCHE, A.; MOTIK, B.; OBERLE, D.; SCHMITZ, C.; STAAB, S.; STOJANOVIC, L.; STOJANOVIC, N.; STUDER, R.; STUMME, G.; SURE, Y.; TANE, J.; VOLZ, R.; ZACHARIAS, V. KAON - Towards a large scale Semantic Web. In: THIRD INTERNATIONAL CONFERENCE ON E-COMMERCE AND WEB TECHNOLOGIES (EC-WEB 2002), AIX-EN-PROVENCE, FRANCE, 2002. **Proceedings...** Springer, 2002. p.304–313. (LNCS, v.2455).

BRICKLEY, D.; GUHA, R. **Resource Description Framework (RDF) Schema Specification**. 1999. (W3C Proposed Recommendation).

BRICKLEY, D.; GUHA, R. **RDF Vocabulary Description Language 1.0**: rdf schema. 2004. (W3C Recommendation).

BRUIJN, J. D.; FRANCONI, E.; TESSARIS, S. Logical reconstruction of normative RDF. In: IN OWL: EXPERIENCES AND DIRECTIONS WORKSHOP (OWLED-2005, 2005. **Anais...** [S.l.: s.n.], 2005.

CHEN, H.; FININ, T.; JOSHI, A. An ontology for context-aware pervasive computing environments. **Knowl. Eng. Rev.**, New York, NY, USA, v.18, p.197–207, September 2003.

CHEN, H.; FININ, T.; JOSHI, A. Semantic Web in the Context Broker Architecture. In: SECOND IEEE INTERNATIONAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS (PERCOM'04), 2004, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2004. p.277–. (PERCOM '04).

CHENG, L.; MARSIC, I. Piecewise network awareness service for wireless/mobile pervasive computing. **Mob. Netw. Appl.**, Hingham, MA, USA, v.7, p.269–278, August 2002.

Ontology Learning and Population from Text: algorithms, evaluation and applications. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

CUSUMANO, M. Cloud computing and SaaS as new computing platforms. **Commun. ACM**, New York, NY, USA, v.53, p.27–29, Apr. 2010.

DEAN, M.; SCHREIBER, G. **OWL Web Ontology Language Reference**. [S.l.]: W3C, 2004. W3C Recommendation.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. **Hum.-Comput. Interact.**, Hillsdale, NJ, USA, v.16, n.2, p.97–166, Dec. 2001.

DEY, A. K.; SALBER, D.; ABOWD, G. D.; FUTAKAWA, M. The Conference Assistant: combining context-awareness with wearable computing. In: IEEE INTERNATIONAL SYMPOSIUM ON WEARABLE COMPUTERS, 3., 1999, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 1999. p.21–. (ISWC '99).

DI FERDINANDO, A.; ROSI, A.; LENT, R.; MANZALINI, A.; ZAMBONELLI, F. MyAds: a system for adaptive pervasive advertisements. **Pervasive Mob. Comput.**, Amsterdam, The Netherlands, The Netherlands, v.5, p.385–401, October 2009.

DOURISH, P. What we talk about when we talk about context. **Personal Ubiquitous Comput.**, London, UK, v.8, p.19–30, February 2004.

FENSEL, D.; HARMELEN, F. van; HORROCKS, I.; MCGUINNESS, D.; PATEL-SCHNEIDER, P. F. OIL: An ontology infrastructure for the semantic web. **IEEE Intelligent Systems**, [S.l.], v.16, n.2, p.38–45, 2001.

FOUQUET, M.; NIEDERMAYER, H.; CARLE, G. Cloud computing for the masses. In: ACM WORKSHOP ON USER-PROVIDED NETWORKING: CHALLENGES AND OPPORTUNITIES, 1., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.31–36. (U-NET '09).

FOURNIER, D.; MOKHTAR, S. B.; GEORGANTAS, N.; ISSARNY, V. Towards ad hoc contextual services for pervasive computing. In: MIDDLEWARE FOR SERVICE ORIENTED COMPUTING (MW4SOC 2006), 1., 2006, New York, NY, USA. **Proceedings...** ACM, 2006. p.36–41. (MW4SOC '06).

FREITAS, L. O. **UMA METODOLOGIA PARA ASSISTIR PACIENTES EM AMBIENTES HOMECARE PERVASIVOS**. 2011. Dissertação (Mestrado) — Programa de Pós-Graduação em Informática, Universidade Federal de Santa Maria, Santa Maria, RS, Brazil.

GASSEN, J. B. **Uma metodologia para o uso de ontologias aplicadas à descrição de contexto em ambientes hospitalares pervasivos**. 2010. Dissertação (Mestrado) — Curso de Mestrado em Nanociências - Centro Universitário Franciscano, Santa Maria, RS, Brazil.

GRUBER, T. R. Toward principles for the design of ontologies used for knowledge sharing. **Int. J. Hum.-Comput. Stud.**, Duluth, MN, USA, v.43, p.907–928, December 1995.

GRUNINGER, M.; LEE, J. Introduction. **Commun. ACM**, New York, NY, USA, v.45, p.39–41, February 2002.

HAZELHURST, S. Scientific computing using virtual high-performance computing: a case study using the amazon elastic computing cloud. In: SOUTH AFRICAN INSTITUTE OF COMPUTER SCIENTISTS AND INFORMATION TECHNOLOGISTS ON IT RESEARCH IN DEVELOPING COUNTRIES: RIDING THE WAVE OF TECHNOLOGY, 2008., 2008, New York, NY, USA. **Proceedings...** ACM, 2008. p.94–103. (SAICSIT '08).

HEFFLIN, J.; VOLZ, R.; DALE, J. **Requirements for a Web Ontology language**. [S.l.: s.n.], 2002.

HENRICKSEN, K.; INDULSKA, J. Developing context-aware pervasive computing applications: models and approach. **Pervasive Mob. Comput.**, Amsterdam, The Netherlands, The Netherlands, v.2, p.37–64, February 2006.

HU, G.; ZHANG, W.; ZHU, W.; SHEN, S. A dynamic user-integrated cloud computing architecture. In: INTERNATIONAL CONFERENCE ON INNOVATIVE COMPUTING AND CLOUD COMPUTING, 2011., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.36–40. (ICCC '11).

JAIN, R.; WULLERT II, J. Challenges: environmental design for pervasive computing systems. In: MOBILE COMPUTING AND NETWORKING, 8., 2002, New York, NY, USA. **Proceedings...** ACM, 2002. p.263–270. (MobiCom '02).

KALAPRIYA, K.; NANDY, S. K.; SRINIVASAN, D.; UMA MAHESHWARI, R.; SATISH, V. A framework for resource discovery in pervasive computing for mobile aware task execution. In: COMPUTING FRONTIERS, 1., 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.70–77. (CF '04).

KOSTAKOS, V. The challenges and opportunities of designing pervasive systems for deep-space colonies. **Personal Ubiquitous Comput.**, London, UK, UK, v.15, p.479–486, June 2011.

LENK, A.; KLEMS, M.; NIMIS, J.; TAI, S.; SANDHOLM, T. What's inside the Cloud? An architectural map of the Cloud landscape. In: ICSE WORKSHOP ON SOFTWARE ENGINEERING CHALLENGES OF CLOUD COMPUTING, 2009., 2009, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2009. p.23–31. (CLOUD '09).

LIANG, X. CRM business cloud computing. In: INTERNATIONAL CONFERENCE ON INNOVATIVE COMPUTING AND CLOUD COMPUTING, 2011., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.103–106. (ICCC '11).

LIBRELOTTO, G. R.; GASSEN, J. B.; FREITAS, L. O.; VIZZOTTO, J. K.; SILVA, F. L.; AUGUSTIN, I. Aplicando o OntoHealth para o Processamento e Consultas em Ontologia para um Ambiente Hospitalar Pervasivo. In: REVISTA BRASILEIRA DE SISTEMAS DE INFORMAçãO, 2009, Rio de Janeiro, Brasil. **Anais...** [S.l.: s.n.], 2009.

LOPES, J. L.; PILLA, M. L.; YAMIN, A. C. EXEHDA: a middleware for complex,heterogeneous and distributed applications. In: IBERIAN-AMERICAN CONFERENCE ON TECHNOLOGY INNOVATION AND STATEGIC AREAS, 2007. **Anais...** [S.l.: s.n.], 2007.

LOUREIRO, E.; OLIVEIRA, L.; ALMEIDA, H. Improving flexibility on host discovery for pervasive computing middlewares. In: MIDDLEWARE FOR PERVASIVE AND AD-HOC COMPUTING, 3., 2005, New York, NY, USA. **Proceedings...** ACM, 2005. p.1–8. (MPAC '05).

LYONS, K.; WANT, R.; MUNDAY, D.; HE, J.; SUD, S.; ROSARIO, B.; PERING, T. Context-aware composition. In: MOBILE COMPUTING SYSTEMS AND APPLICATIONS, 10., 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.12:1–12:6. (HotMobile '09).

MAGERKURTH, C.; MEMISOGLU, M.; ENGELKE, T.; STREITZ, N. Towards the next generation of tabletop gaming experiences. In: GRAPHICS INTERFACE 2004, 2004, School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada. **Proceedings...** Canadian Human-Computer Communications Society, 2004. p.73–80. (GI '04).

MALEK, S.; SEO, C.; RAVULA, S.; PETRUS, B.; MEDVIDOVIC, N. Reconceptualizing a Family of Heterogeneous Embedded Systems via Explicit Architectural Support. In: SOFTWARE ENGINEERING, 29., 2007, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2007. p.591–601. (ICSE '07).

MCGUINNESS, D.; HARMELEN, F. van. **OWL Web Ontology Language Overview**. [S.l.]: W3C, 2006. W3C Recommendation.

MEDVIDOVIC, N.; MALEK, S. Software deployment architecture and quality-of-service in pervasive environments. In: INTERNATIONAL WORKSHOP ON ENGINEERING OF SOFTWARE SERVICES FOR PERVASIVE ENVIRONMENTS: IN CONJUNCTION WITH THE 6TH ESEC/FSE JOINT MEETING, 2007, New York, NY, USA. **Anais...** ACM, 2007. p.47–51. (ESSPE '07).

MELL, P.; GRANCE, T. The NIST Definition of Cloud Computing. **National Institute of Standards and Technology**, [S.l.], p.7, 2011.

MILLER, T.; MCBURNEY, P. A formal semantics for Gaia liveness rules and expressions. **Int. J. Agent-Oriented Softw. Eng.**, Inderscience Publishers, Geneva, SWITZERLAND, v.1, p.435–476, December 2007.

NAPPER, J.; BIENTINESI, P. Can cloud computing reach the top500? In: UNCONVENTIONAL HIGH PERFORMANCE COMPUTING WORKSHOP PLUS MEMORY ACCESS WORKSHOP, 2009, New York, NY, USA. **Proceedings...** ACM, 2009. p.17–20. (UCHPC-MAW '09).

NOY, N. F.; MCGUINNESS, D. L. **Ontology Development 101**: a guide to creating your first ontology. [S.l.]: Stanford Knowledge Systems Laboratory and Stanford Medical Informatics, 2001.

O'SULLIVAN, D.; LEWIS, D. Semantically driven service interoperability for pervasive computing. In: ACM INTERNATIONAL WORKSHOP ON DATA ENGINEERING FOR WIRELESS AND MOBILE ACCESS, 3., 2003, New York, NY, USA. **Proceedings...** ACM, 2003. p.17–24. (MobiDe '03).

PAGANELLI, F.; BIANCHI, G.; GIULI, D. A context model for context-aware system design towards the ambient intelligence vision: experiences in the etourism domain. In: USER IN-

TERFACES FOR ALL, 9., 2007, Berlin, Heidelberg. **Proceedings...** Springer-Verlag, 2007. p.173–191. (ERCIM'06).

PAN, Y.; BLEVIS, E. The cloud. **interactions**, New York, NY, USA, v.18, p.13–16, January 2011.

PHAM, T. V.; JAMJOOM, H.; JORDAN, K.; SHAE, Z.-Y. A service composition framework for market-oriented high performance computing cloud. In: ACM INTERNATIONAL SYMPOSIUM ON HIGH PERFORMANCE DISTRIBUTED COMPUTING, 19., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.284–287. (HPDC '10).

RANGANATHAN, A.; AL-MUHTADI, J.; CHETAN, S.; CAMPBELL, R.; MICKUNAS, M. D. MiddleWhere: a middleware for location awareness in ubiquitous computing applications. In: ACM/IFIP/USENIX INTERNATIONAL CONFERENCE ON MIDDLEWARE, 5., 2004, New York, NY, USA. **Proceedings...** Springer-Verlag New York: Inc., 2004. p.397–416. (Middleware '04).

RANGANATHAN, A.; CAMPBELL, R. H. Advertising in a pervasive computing environment. In: MOBILE COMMERCE, 2., 2002, New York, NY, USA. **Proceedings...** ACM, 2002. p.10–14. (WMC '02).

REESE, G. Cloud Computing. In: **Cloud Application Architectures**: building applications and infrastructure in the cloud. [S.l.]: O'Reilly Media, Inc., 2009.

RELLERMEYER, J. S.; DULLER, M.; ALONSO, G. Engineering the cloud from software modules. In: ICSE WORKSHOP ON SOFTWARE ENGINEERING CHALLENGES OF CLOUD COMPUTING, 2009., 2009, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2009. p.32–37. (CLOUD '09).

ROBINSON, J.; WAKEMAN, I.; OWEN, T. Scooby: middleware for service composition in pervasive computing. In: MIDDLEWARE FOR PERVASIVE AND AD-HOC COMPUTING, 2., 2004, New York, NY, USA. **Proceedings...** ACM, 2004. p.161–166. (MPAC '04).

SCHILIT, B. N.; HILBERT, D. M.; TREVOR, J. Context-aware communication. **Wireless Communications, IEEE [see also IEEE Personal Communications]**, [S.l.], v.9, n.5, p.46–54, 2002.

SOLDATOS, J.; DIMAKIS, N.; STAMATIS, K.; POLYMENAKOS, L. A breadboard architecture for pervasive context-aware services in smart spaces: middleware components and prototype applications. **Personal Ubiquitous Comput.**, London, UK, UK, v.11, p.193–212, February 2007.

SOSINSKY, B. Defining Cloud Computing. In: **Cloud Computing Bible**. [S.l.]: John Wiley & Sons, 2011. (Bible Series).

SOWA, J. F. Ontology, Metadata, and Semiotics. In: LINGUISTIC ON CONCEPTUAL STRUCTURES: LOGICAL LINGUISTIC, AND COMPUTATIONAL ISSUES, 2000, London, UK. **Proceedings...** Springer-Verlag, 2000. p.55–81.

SUMTER, L. Cloud computing: security risk. In: ANNUAL SOUTHEAST REGIONAL CONFERENCE, 48., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.112:1–112:4. (ACM SE '10).

TAKAHASHI, T.; KADOBAYASHI, Y.; FUJIWARA, H. Ontological approach toward cybersecurity in cloud computing. In: SECURITY OF INFORMATION AND NETWORKS, 3., 2010, New York, NY, USA. **Proceedings...** ACM, 2010. p.100–109. (SIN '10).

TRAN, V.; KEUNG, J.; LIU, A.; FEKETE, A. Application migration to cloud: a taxonomy of critical factors. In: INTERNATIONAL WORKSHOP ON SOFTWARE ENGINEERING FOR CLOUD COMPUTING, 2., 2011, New York, NY, USA. **Proceedings...** ACM, 2011. p.22–28. (SECLOUD '11).

VELTE, A.; VELTE, T. J.; ELSENPETER, R. Cloud Computing Basics. In: **Cloud Computing**: a pratical approach. [S.l.]: McGraw-Hill Prof Med/Tech, 2009.

WANG, X. H.; ZHANG, D. Q.; GU, T.; PUNG, H. K. Ontology Based Context Modeling and Reasoning using OWL. In: SECOND IEEE ANNUAL CONFERENCE ON PERVASIVE COMPUTING AND COMMUNICATIONS WORKSHOPS, 2004, Washington, DC, USA. **Proceedings...** IEEE Computer Society, 2004. p.18–. (PERCOMW '04).

WANT, R.; PERING, T. System challenges for ubiquitous & pervasive computing. In: SOFTWARE ENGINEERING, 27., 2005, New York, NY, USA. **Proceedings...** ACM, 2005. p.9–14. (ICSE '05).

WEISER, M. The computer for the 21st century. **Scientific American**, [S.l.], September 1991.

XU, C.; CHEUNG, S. C.; CHAN, W. K.; YE, C. Partial constraint checking for context consistency in pervasive computing. **ACM Trans. Softw. Eng. Methodol.**, New York, NY, USA, v.19, p.9:1–9:61, February 2010.

YAMIN, A. C. **Arquitetura para um Ambiente de Grade Computacional Direcionado às Aplicações Distribuídas, Móveis e Conscientes do Contexto da Computação Pervasiva**. 2004. 195p. Tese de Doutorado — II/UFRGS, Porto Alegre, RS, Brasil.

YE, J.; COYLE, L.; DOBSON, S.; NIXON, P. Ontology-based models in pervasive computing systems. **Knowl. Eng. Rev.**, New York, NY, USA, v.22, p.315–347, December 2007.