Aplicando o OntoHealth para o Processamento e Consultas em Ontologia para um Ambiente Hospitalar Pervasivo

Giovani Rubert Librelotto[1], Jonas Bulegon Gassen[2], Leandro O. Freitas[2], Juliana K. Vizzotto[1], Fabio Lorenzi da Silva[1] e Iara Augustin[1]

[1] UFSM – Universidade Federal de Santa Maria
DELC – Departamento de Eletrônica e Computação
Av. Roraima, 1000, Bairro Camobi, 97105-900, Santa Maria - RS, Brasil

[2]UNIFRA – Centro Universitário Franciscano
Rua dos Andradas, 1614, Santa Maria - RS, 97010-032, Brasil

{librelotto,augustin}@inf.ufsm.br

Abstract. *This paper presents a system to process ontologies applied to pervasive environment. The main idea is that a hospital can be seen as this pervasive environment, where someone "using" ubiquitous computing engages many computational devices and systems simultaneously, in the course of ordinary activities, and may not necessarily even be aware that they are doing so. With this proposed ontology and the tool for this processing, the medical tasks can be shared by all components of this pervasive environment.*

Resumo. *Um ambiente hospitalar pervasivo necessita que as entidades presentes neste contexto (como pessoas e equipamentos) estejam em perfeita sincronia para a realização das tarefas médicas comuns no seu dia-a-dia. Uma das maneiras mais indicadas para efetuar a representação deste domínio é através de ontologias. Desta forma, este trabalho tem como principal objetivo mostrar o processo de desenvolvimento de uma ferramenta para o processamento de ontologias em ambientes pervasivos hospitalares.*

1. Introdução

O sistema de saúde do futuro prevê o uso de tecnologias da Computação Pervasiva [Saha and Mukherjee 2003] formando um espaço inteligente, reativo e pró-ativo, onde elementos computacionais tomarão decisões e adaptar-se-ão às situações detectadas. Esse novo sistema de saúde também prevê uma visão de hospital virtual (*HealthyHome*), o qual estende-se para a casa dos pacientes ou lugares onde eles se encontram, onde sensores/dispositivos monitoram as condições ambientais e do paciente e comunicam-se, via rede sem fio, com as centrais médicas para tomada de decisões e ações pertinentes [Chen and Finin 2003]. Experiências nesse sentido estão sendo conduzidas por alguns projetos de pesquisa, como o do *Centre of Pervasive Healthcare* na Dinamarca [Laerum and Faxvaag 2004]. Neste momento, em termos de pesquisa e inovação, a Computação Pervasiva na Saúde está em sua primeira geração, a qual procura entender as necessidades, características e tecnologias para projetar sistemas que criarão o hospital do futuro. No Brasil, observa-se que o maior foco das pesquisas em Informática na Saúde é ainda em registros de pacientes e medicamentos, prontuário eletrônico e informação sobre cursos clínicos.

A computação pervasiva requer que as tarefas computacionais estejam cientes dos ambientes adjacentes e das necessidades dos usuários além da capacidade de adaptação a estes. Uma das noções fundamentais na computação pervasiva é a sensibilidade ao contexto. Por contexto entende-se qualquer informação relevante que pode ser utilizada para caracterizar a situação de alguma entidade. Por sua vez, a computação sensível ao contexto define-se pelo uso de características do ambiente, tais como a localização do usuário, tempo e atividade para permitir que aplicações adaptem-se as situações e forneçam informações relevantes ao usuário. Desta forma, as informações referentes ao contexto devem ser relacionadas com a representação do conhecimento do domínio em questão. Uma das maneiras mais adequadas de representação de conhecimento é através do uso de ontologias.

Em uma ontologia, os relacionamentos são definidos formalmente e a semântica de um dado relacionamento é detalhada. Se esses relacionamentos possuem certos nomes apropriados que identificam seu significado, um humano pode entendê-la diretamente; assim como um programa pode assumir a semântica de um dado relacionamento e atuar sistematicamente através da mesma.

O processo de construção de ontologias não é uma tarefa trivial, tendo em vista que para a definição da mesma é necessário um conhecimento especializado de forma a não haver qualquer tipo de ambigüidade e contestações quanto a sua validade. Sendo assim, este artigo tem como objetivo principal a criação de uma ontologia para descrever o domínio de conhecimento de um hospital, de modo que esta ontologia possa ser utilizada para a interação entre as entidades em um ambiente pervasivo, e a implementação de um sistema que permita seu processamento.

Este artigo é uma versão estendida do artigo publicado no SBSI 2009 [Librelotto et al. 2009]. Além disso, neste artigo encontram-se os resultados obtidos a partir das pesquisas que geram outros trabalhos publicados em eventos pelos autores, como [Ferreira et al. 2009] e [Librelotto et al. 2008].

De forma a introduzir os conceitos básicos, a seção 2 apresentará os conceitos relacionados à Computação Pervasiva e a relação destes com um ambiente hospitalar. A seção 3 introduzirá as ontologias e sua utilidade neste contexto. Em seguida, será discutida a metodologia do projeto, onde a seção 4 descreve a criação de uma ontologia para um hospital como ambiente pervasivo e a seção 6 apresenta o modo de processamento de ontologias hospitalares. A seção 9 descreve a conclusão do trabalho, após a referência aos trabalhos relacionados na seção 8.

2. Um hospital como um ambiente pervasivo

A computação pervasiva tem por objetivo disponibilizar informações e recursos aos usuários a qualquer hora e em qualquer lugar [Saha and Mukherjee 2003]. Em um ambiente pervasivo, o computador está disposto de forma transparente ao usuário, de forma que este não percebe que está utilizando um computador para realizar suas tarefas. Portanto, o computador deve ter a capacidade de obter informações deste ambiente e utilizá-las para construir modelos computacionais dinamicamente, ou seja, controlar, configurar e ajustar a aplicação para melhor atender as necessidades do usuário.

Além disto, o ambiente também deve ser capaz de detectar outros dispositivos que venham a fazer parte dele e que possam ser relevantes no auxilio de realização de tarefas.

Desta interação surge a capacidade de computadores agirem de forma "inteligente" em um ambiente povoado por sensores e serviços computacionais [Chen and Finin 2003].

Um hospital pode ser visto como um ambiente pervasivo quando os seus dispositivos computacionais passarem a interagir com os usuários de forma tão natural a ponto que tais dispositivos sejam considerados invisíveis.

Os dispositivos encontrados em um ambiente pervasivo hospitalar devem ser sensíveis ao contexto, ou seja, devem trabalhar de forma dinâmica, realizando freqüentes varreduras no ambiente para verificar as mudanças que ocorrem nele, adaptando-se e atualizando-se constantemente. A adaptação é muito importante, pois um ambiente hospitalar pervasivo pode ser bastante heterogêneo, possuindo uma grande variedade de dispositivos com diferentes plataformas de processamento.

De modo a garantir a interoperabilidade sobre o conhecimento entre os dispositivos encontrados neste ambiente pervasivo, optou-se pela representação do domínio em uma ontologia. A idéia central do artigo é mostrar como é feito o processamento de uma ontologia criada para um ambiente pervasivo hospitalar. Nas próximas seções descreve-se como as ontologias são úteis, como definiu-se este ambiente e como é realizado tal processamento.

3. Ontologias

Ontologias podem ser vistas como um conjunto coerente de coleções estruturadas de informação. Uma ontologia é uma teoria lógica para descrever o significado pretendido de um vocabulário formal, isto é, seu compromisso com uma conceitualização particular do mundo. Estas incluem estruturas que permitem manipular termos de uma forma muito eficiente e útil para o usuário e mecanismos de validação para comunicação entre programas. A importância de seu uso é devida à capacidade de representar hierarquias de classes de objetos (taxonomias) e seus relacionamentos.

Guarino define a ontologia como uma caracterização axiomática do significado do vocabulário lógico [Guarino 1997]. Na área de Sistemas de Informação, ontologia é definida como um conjunto de conceitos e termos que podem ser usados para descrever uma representação para o conhecimento [Swartout and Tate 1999].

Segundo Chandrasekaran [Chandrasekaran 1999], ontologias são teorias de conteúdo sobre os tipos de objetos, propriedades de objetos e relacionamentos entre objetos que são possíveis num domínio de conhecimento específico.

Desta forma, uma ontologia pode ser utilizada em um ambiente hospitalar pervasivo, de forma que a mesma sirva de base para a interoperabilidade entre os dispositivos móveis e fixos, pois a ontologia define as classes e suas propriedades que existem no domínio desta aplicação e relacionamentos entre suas instâncias. Para isso, a ontologia deve ser dinâmica para que ela possa fornecer restrições de acesso aos dados e recursos neste ambiente, além da interação entre dispositivos/dispositivos e dispositivos/médicos.

As ontologias podem ser expressas em linguagens como RDF e OWL, as quais serão apresentadas a seguir.

3.1. RDF

RDF é a sigla para *Resource Description Framework*, e é uma recomendação da W3C para representação de informações [Horrocks 2008]. Seus principais objetivos são criar modelos simples de dados; usar o vocabulário *URI-based*; e dar suporte ao uso de XML. Esta linguagem possui três componentes básicos: recurso, propriedade e indicação. Um recurso é qualquer coisa que possa conter um URI (como páginas da Web) e elementos de um arquivo XML. As propriedades têm a função de identificar as classes e também definir ações que podem ser exercidas pelas mesmas. Já uma indicação consiste na combinação de um recurso, uma propriedade e um valor.

3.2. OWL

A linguagem OWL (*Web Ontology Language*) foi projetada para o uso de aplicações que precisam processar o conteúdo de uma informação, e não apenas apresentá-la [Mcguineness and Harmelen 2004]. Como possui um dicionário adicional e uma semântica formal, possibilita uma melhor expressividade semântica do que a disponível em XML e RDF. A linguagem OWL [Mcguineness and Harmelen 2004] é uma linguagem para definir e instanciar ontologias, fornecendo uma especificação que permite representar conhecimento conceitual com o qual se distingue recurso de informações semanticamente.

Esta linguagem possui base na lógica de descrição e pode ser definida em três sub-linguagens, cada uma correspondente a uma lógica de descrição.

A OWL Lite possui uma complexidade menor do que as outras duas sub-linguagens, sendo utilizada por usuários que apenas necessitam de uma classificação hierárquica e restrições simples. OWL DL possui correspondência com as lógicas de descrição. É indicada aos usuários que precisam de máxima expressividade e com a garantia de processamento computacional. Ao contrário da OWL DL, a OWL Full não oferece garantia computacional, podendo nunca obter a informação desejada no processamento de uma ontologia, porém possui a liberdade de sintaxe do RDF [Horrocks 2008]. A OWL Full mesmo com sua grande expressividade e liberdade de sintaxe, possui contratempos como um maior tempo para processamento da informação, devido a esta liberdade oferecida ao programador na hora de escrever um código.

4. Especificação de uma ontologia para um ambiente hospitalar pervasivo

As necessidades encontradas em um hospital foram relativas às atividades realizadas pelos clínicos, portanto foi feita a especificação de uma ontologia para tarefas médicas. Esta seção propõe um modelo ontológico que possa servir de base para implementação de modelos computacionais de mais alto nível direcionados à Computação Pervasiva. Com o objetivo de modelar as tarefas clínicas e tratar a diversidade de atividades executadas pelos profissionais, adotou-se a classificação de tarefas baseada na proposta em [Kumar et al. 2003].

Uma tarefa pode ser qualquer atividade, ou seqüencia de atividades, que envolvam recursos computacionais, realizada por um clinico em relação a um paciente. Define-se tarefa como o conjunto de ações executadas colaborativamente por humanos e sistemas de computação pervasivos. As tarefas podem ser compostas por sub-tarefas. Tarefas agrupadas compõem um fluxo de trabalho. As tarefas ainda são assistidas por aplicações com-

putacionais. As sub-tarefas representam abstrações de serviços a serem disponibilizados aos médicos para que, a partir dessas, estes possam modelar as tarefas desejadas.

Estas tarefas são dinâmicas, ou seja, a partir de um grupo mínimo de sub-tarefas, as quais são constituídas por atividades, podem ser criadas novas tarefas. Em resumo, tarefa é um conjunto que contém sub-tarefas ou atividades médicas. Pode-se fazer uma analogia a um sistema de arquivos, composto por pastas e arquivos, onde as pastas seriam tarefas e arquivos seriam atividades.

O conjunto mínimo de tarefas foi definido baseado no levantamento realizado com médicos para a identificação e definição das principais tarefas clínicas executadas pelos profissionais dessa área [Laerum and Faxvaag 2004]. O estudo propõe um conjunto de vinte e quatro (24) atividades realizadas em ambientes hospitalares, baseado em aspectos como a relevância da atividade e periodicidade com que a mesma é executada.

Para a especificação da ontologia, utilizou-se as atividades executadas com maior frequência no dia-a-dia, chegando-se ao total de onze (11) tarefas clínicas, as quais compõem o conjunto mínimo de tarefas. Ressalta-se que o conjunto mínimo de tarefas foca unicamente o trabalho realizado por médicos, sendo esse justamente o público-alvo ao qual se destina esta ontologia. Chegou-se a estas 11 tarefas a partir de uma consulta com médicos dos hospitais universitários da UCPel, UFPel e UFSM.

A visão da rede semântica que representa as tarefas e sub-tarefas encontradas na ontologia encontra-se na figura 1.

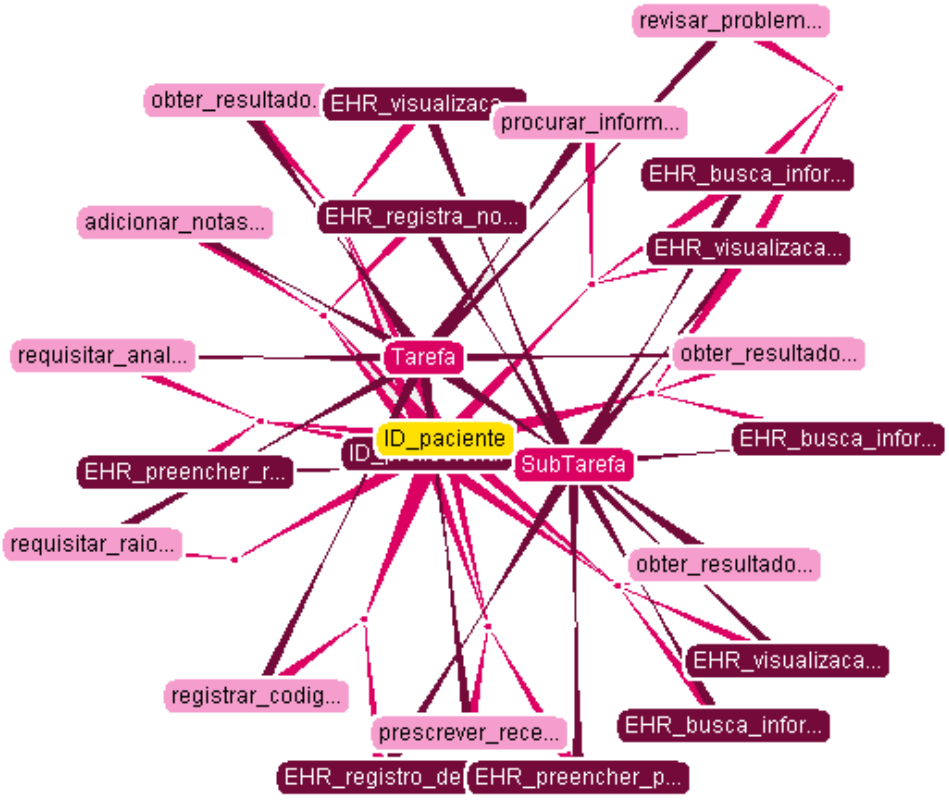

Figura 1. Visão da rede semântica representada na ontologia

Na figura 1 percebe-se a relação entre as as tarefas e as sub-tarefas (iniciadas por ID ou EHR) que as compõem. Cada ligação indica uma relação específica. A maioria das sub-tarefas compõe mais de uma tarefa, o que indica uma nova ligação, gerando tal emaranhado de relações.

Um sistema EHR (*Electronic Health Record*) deve gerenciar e armazenar as informações sobre a saúde dos pacientes com o acréscimo de aspectos de pervasividade. Assim, este deve apresentar suporte a funcionalidades características da Computação Pervasiva como migração e restauração da sessão do usuário e adaptação aos diversificados dispositivos que o usuário pode utilizar para interagir com o sistema.

De forma a simplificar a visualização da ontologia, a figura 2 apresenta apenas as relações envolvendo a tarefa "*revisar problemas do paciente*". Esta é do tipo "*tarefa*" e é composta por quatro sub-tarefas. Neste caso em particular, as sub-tarefas a serem executadas no âmbito desta tarefa seriam: a identificação do profissional que vai realizar a tarefa, a identificação do paciente a ser consultado, a busca das informações do paciente junto ao EHR e a visualização das informações buscadas.

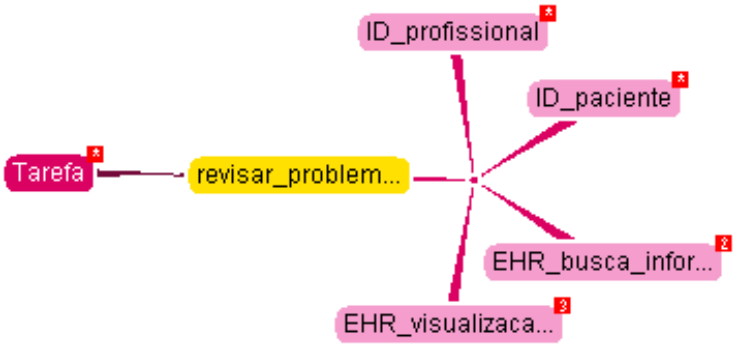

Figura 2. Rede semântica da tarefa "*revisar problemas do paciente*"

Como pode-se perceber, uma tarefa médica é composta por sub-tarefas, as quais são abstrações de serviços existentes em um ambiente hospitalar. Em um nível mais amplo, encontram-se os fluxos. Os fluxos são formados por tarefas agrupadas, as quais serão executadas em uma seqüência simples. Eles devem obedecer a uma ordem de execução previamente definida, impedindo que fluxos impossíveis de serem realizados sejam criados como, por exemplo, um fluxo onde a tarefa "*obter resultados laboratoriais*"esteja localizada antes da tarefa "*requisitar análises laboratoriais*". A figura 3 apresenta um fluxo composto por quatro tarefas: "*revisar problemas do paciente*", "*requisitar análises laboratoriais*", "*obter resultados laboratoriais*" e "*escrever prescrições*", criado por um médico referente ao tratamento de um paciente.

Partindo das onze tarefas básicas inseridas na ontologia, e das sub-tarefas que as compõem, os médicos poderão construir novas tarefas e novos fluxos de execução, conforme a sua preferência e necessidade, a partir de uma interface especializada. A ontologia criada é, por fim, representada em OWL.

5. A construção da ontologia para tarefas médicas

O principal propósito de uma ontologia hospitalar é tornar explícita a informação de maneira independente das estruturas de dados subjacentes que podem ser usadas para

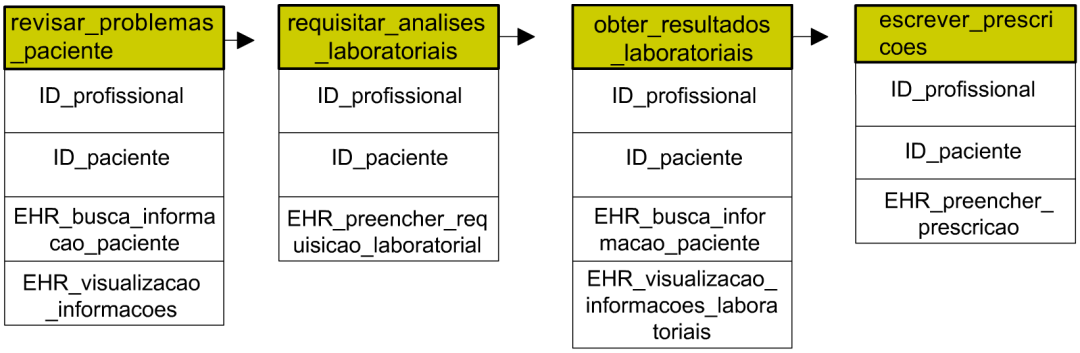

Figura 3. Exemplo de uma tarefa e suas sub-tarefas

armazenar a informação em repositório de dados. As ontologias são abstrações e podem descrever diferentes tipos de organização de dados como tabelas relacionais, textos e imagens. Os usuários (clínicos) deveriam ser capazes de interagir com a ontologia ao invés de interagir com vários depósitos de dados heterogêneos. Assim, os clínicos formulariam consultas sobre a ontologia e o sistema teria a responsabilidade de gerenciar a heterogeneidade e distribuição nos depósitos, isto é, uma ontologia define uma linguagem que será usada para formular consultas. Desta forma, a ontologia criada neste contexto será utilizada para que se possa, além de padronizar os termos médicos utilizados em um ambiente hospitalar, permitir o uso da mesma pelos diversos dispositivos heterogêneos que estão espalhados por este ambiente, possibilitando também a realização de inferências sobre os dados dos pacientes.

A ontologia criada para auxiliar médicos e enfermeiros em suas tarefas foi dividida em classes e propriedades.

Sobre modelagem das classes, é importante salientar algumas informações. A classe *Pessoa* foi definida como uma classe abstrata, pois não poderá haver nenhum indivíduo pertencente apenas a esta classe. Ela será superclasse das classes: *Clínico* e *Paciente*. A classe *Clínico* por sua vez, será superclasse das classes *Médico* e *Enfermeiro*. Para todas as subclasses de *Pessoa* foram atribuídas duas propriedades em comum: nome e idade. Na figura 4 é apresentada uma visão geral da ontologia e a hierarquia entre elas.

Figura 4. Hierarquia das classes

As propriedades têm a função de identificar as classes e também definir ações que podem ser exercidas pelas mesmas. Como exemplo de uma propriedade de dados, podemos citar a propriedade nome, que representa o nome de um indivíduo de qualquer uma das subclasses de *Pessoa*. Para exemplificar uma propriedade que define uma ação correspondente à uma determinada classe, pode-se usar a propriedade receitar, a qual refere-se

a ação de um indivíduo da classe *Médico* que receita um (ou mais) medicamentos.

A classe *Médico* é uma subclasse de *Pessoa*, portanto ela herda de sua superclasse as propriedades nome e idade. A figura 5 mostra a relação da classe *Médico* e suas propriedades exclusivas.

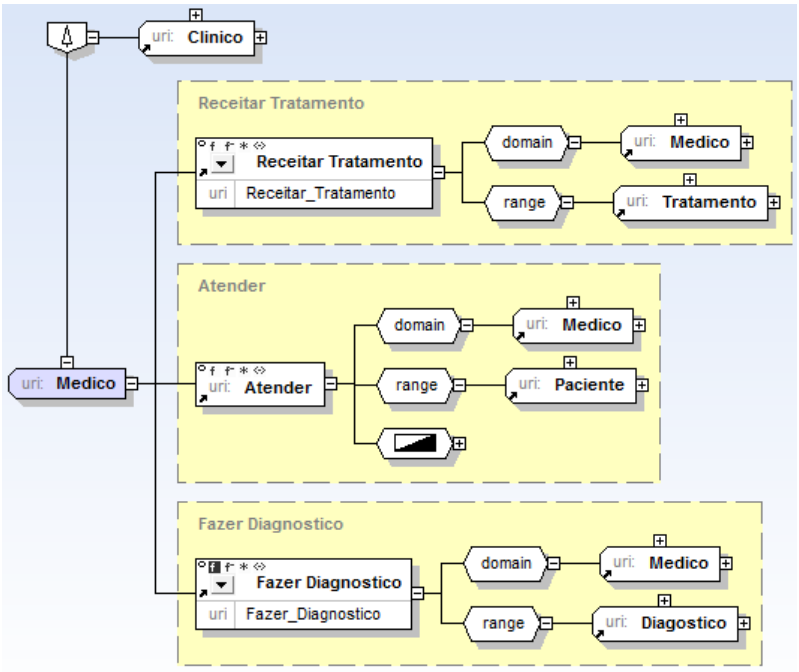

Figura 5. Grafo Médico x Propriedades

O trecho de código a seguir mostra a definição da classe Médico.

```
<owl:Class rdf:ID="Medico">
   <rdfs:subClassOf rdf:resource="#Clinico"/>
</owl:Class>
```

Em OWL, o elemento *owl:ObjectProperty* indica a criação de uma propriedade que relaciona duas classes [Fensel et al. 2001]. No caso da propriedade atender, elemento *rdfs:domain* identifica a quem pertence esta propriedade, enquanto o elemento *rdfs:range* identifica a classe que um médico irá atender.

O código a seguir apresenta a definição da propriedade atender, onde o domínio da propriedade é qualquer indivíduo da classe Médico e o limite da propriedade pode ser qualquer indivíduo da classe Paciente.

```
<owl:ObjectProperty rdf:about="#Examina">
   <rdfs:range rdf:resource="#Paciente"/>
   <rdfs:domain rdf:resource="#Medico"/>
   <owl:inverseOf rdf:resource="#Examinado_por"/>
</owl:ObjectProperty>
```

Além destas propriedades previamente criadas, a ontologia conta também com a criação dinâmica de novas tarefas que podem conter "n" tarefas já existentes. Se um médico deseja realizar uma tarefa que não consta na lista de tarefas definidas, ele pode selecionar, por exemplo, as tarefas *examina_paciente* e *solicita_raioX* para formar uma nova tarefa. Esta criação dinâmica de tarefas é feita através do *OntoHealth*.

6. OntoHealth – Um sistema para o processamento de ontologias hospitalares

A fim de garantir a pervasividade e o acesso as informações contidas no ambiente, faz-se necessário o processamento da ontologia acima definida. Para este fim, criou-se o OntoHealth. O OntoHealth é um *framework* que permite o processamento da ontologia desenvolvida para um ambiente hospitalar pervasivo descrita neste artigo.

O acesso à ontologia é realizado através de interfaces específicas a partir de dispositivos móveis (como PDAs e *smartphones*) ou de dispositivos fixos que estão inseridos no ambiente. O OntoHealth possui métodos que possibilitam a criação de interfaces para a manipulação da ontologia. Esses métodos permitem, entre outras funções, a realização de consultas na ontologia, a criação de novas tarefas e de novos fluxos e a exportação da ontologia para algum formato de arquivo (como OWL e Topic Maps [Librelotto et al. 2006]).

De acordo com a arquitetura apresentada na figura 6, os dispositivos acessam o conhecimento expresso na ontologia através de interfaces ou programas, as quais se comunicam com aplicações desenvolvidas com o sistema OntoHealth, responsável pelo processamento (armazenamento, consulta e atualização) desta ontologia.

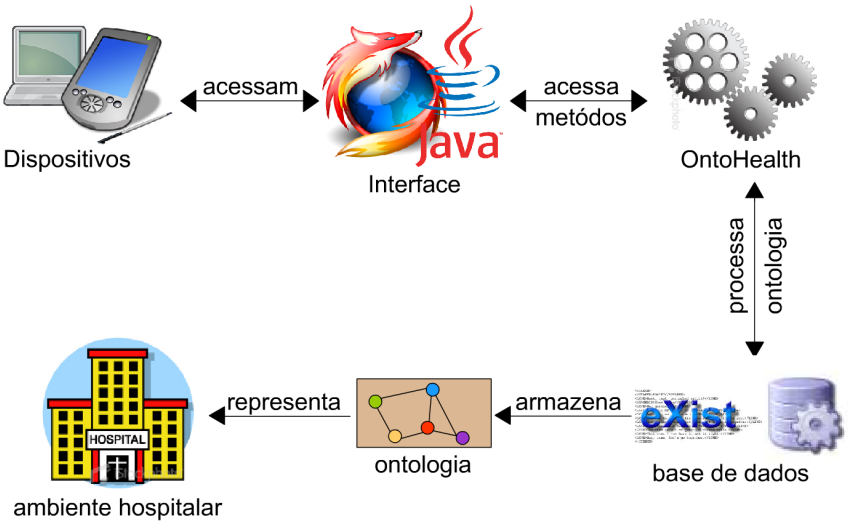

Figura 6. A arquitetura do sistema

6.1. Métodos do Ontohealth

O acesso à ontologia é realizado através de interfaces específicas a partir de dispositivos móveis (como PDAs e smartphones) ou de dispositivos fixos inseridos no ambiente. Todo e qualquer processamento referente à ontologia, seja em relação a banco de dados ou aplicações internas do sistema será feito através do OntoHealth.

Este sistema possui métodos que permitem, entre outras coisas, a realização de consultas na ontologia, a criação de novas tarefas e fluxos de trabalho. Os métodos do OntoHealth que possibilitam consultas na ontologia utilizam, de forma transparente ao dispositivo, as linguagens XPath, XQuery e SQWRL. Desta forma, o dispositivo e seu usuário não necessitam dominar tais linguagens, pois as mesmas serão abstraídas pelo método chamado. A seguir estão todos os métodos que constituem o sistema.

- *parseName*;
- *setCollectionPath*;
- *getCollectionPath*;
- *countSubTasks*;
- *countTasks*;
- *countWorkFlows*;
- *getTask*;
- *getSubTasks*;
- *getTasks*;
- *getWorkFlows*;
- *getOntology*;
- *taskExists*;
- *addTask*;
- *addWorkFlow*.

O método *parseName* recebe por parâmetro o nome da tarefa criada pelo médico e altera seu nome deixando-o dentro dos padrões usados na ontologia. Por exemplo, se um médico criar uma tarefa para requisitar um procedimento cirúrgico, o nome da tarefa será "Requisitar procedimento cirúrgico" e é recebido por parâmetro pelo método que irá transformar este nome. O retorno será no formato "requisitar_procedimento_cirurgico". Por sua vez, o método *setCollectionPath* tem a função de definir o caminho da coleção, que será recebida por parâmetro.

O método *getCollectionPath* é usado para retornar o caminho da coleção. Cada tarefa da ontologia possui um número que corresponde ao seu código identificador. Para garantir que uma tarefa, ao ser criada, não possua o mesmo identificador de uma tarefa já existente na ontologia, foi definido o método *countTasks* cujo objetivo é contar o número de tarefas que a ontologia possui. Este método é invocado cada vez que uma nova tarefa é criada e irá retornar o total de tarefas da ontologia. Assim, o código identificador da nova tarefa corresponde ao resultado retornado neste método mais (+) um. Com este mesmo objetivo, também foi criado o método *countWorkFlows* usado para contar o número de fluxos de trabalho existentes na ontologia, que será invocado cada vez que um novo fluxo for criado.

Atualmente os médicos não podem criar novas sub-tarefas. Elas são pré-definidas na especificação do sistema. Contudo, de forma a permitir a atualizações futuras, definiu-se o método *countSubTasks* para contar o número de sub-tarefas da ontologia. Este método tem o mesmo objetivo dos outros dois citados logo acima. Ele não influenciará no desempenho do sistema, já que não será invocado em momento algum.

O método *getTask* tem por objetivo realizar a busca de uma determinada tarefa na ontologia. Este método recebe por parâmetro o identificador e o tipo da tarefa e retornará os detalhes da tarefa, caso ela exista. Este método é usado também para busca de uma sub-tarefa ou fluxo específico. O método *getSubTasks*, possibilita a busca pelas sub-tarefas da ontologia. Quando este método é invocado, é retornado ao usuário todas as sub-tarefas inseridas na ontologia. Com o mesmo objetivo, foram criados os métodos *getTasks* e *getWorkFlows*, que retorna o total de tarefas e fluxos de trabalho respectivamente.

A fim de possibilitar a realização de *backups* da ontologia, foi criado o método *getOntology*, que é usado para retornar todo o conteúdo existente na mesma. O método

taskExists recebe parâmetro o código identificador e o tipo de uma determinada tarefa e realiza uma busca na ontologia, retornando um valor booleano que dirá se ela existe ou não. Este método pode ser usado para que um médico possa verificar se uma determinada tarefa existe antes de criá-la. O método *addTask* é usado para a criação de novas tarefas. Este método irá receber por parâmetro os identificadores das sub-tarefas que farão parte da nova tarefa, montando-a e adicionando-a a ontologia. Com o mesmo objetivo, foi definido o método *addWorkFlow* usado para criação de fluxos de trabalho, que será descrito detalhadamente a seguir

6.2. Funcionamento dos métodos

Esta seção tem por objetivo mostrar o funcionamento dos métodos criados para o sistema OntoHealth através de uma explicação detalhada do método *addWorkFlow* que será usado como exemplo. Conforme descrito no anteriormente, as tarefas podem ser formadas apenas por sub-tarefas e/ou tarefas, mas nunca irá possuir fluxos de trabalho. Por isso, o método do sistema que possibilita a construção de novas tarefas, possui restrições que impossibilitam o médico de adicionar tarefas ou fluxos de trabalho na criação de uma tarefa.

Neste método os identificadores das sub-tarefas, tarefas e fluxos de trabalho que serão usados na criação do novo fluxo são recebidos por parâmetro. Então são criadas três variáveis do tipo *String*:

- xqueryS (para sub-tarefas);
- xqueryT (para tarefas);
- xqueryF (para fluxos de trabalho).

A seguir, cria-se uma string de consulta utilizando a linguagem XQuery para, posteriormente, buscar as sub-tarefas que serão adicionadas ao fluxo que está sendo criado na sub-ontologia sub-tarefas.owl. Esta string de consulta ficará armazenada na variável *xqueryS*. O mesmo processo é realizado para criar strings de consulta aos documentos tarefas.owl e fluxos.owl, que serão armazenadas nas variáveis *xqueryT* e *xqueryF*, respectivamente.

O próximo passo é fazer a conexão com o banco de dados para, então, buscar a coleção que contém os arquivos da ontologia. Feito isso, a *string* de consulta armazenada na variável *xqueryS* é executada e as sub-tarefas retornadas são adicionadas ao novo fluxo entre as *tags* <composto_por> e </composto_por>. O mesmo processo é realizado para adicionar as tarefas e fluxos retornados das consultas executadas com as variáveis *xqueryT* e *xqueryF*.

O último passo é executar uma *query* XUpdate para adicionar ao fluxo criado à coleção. Com o passar do tempo o volume de informações presentes na ontologia tende a crescer consideravelmente devido a criação de novas tarefas e fluxos de trabalho.

Assim, a integração da ontologia com um sistema gerenciador de banco de dados (SGDB) tornou-se indispensável. Desta forma, optou-se pela utilização do SGBD nativo XML eXist [Meier 2008]. O eXist foi desenvolvido em Java e tem seu código aberto, possibilitando assim a criação de métodos que possam ser acessados por interfaces (ou programas) que manipulem as ontologias OWL diretamente no SGBD. Desta forma, usa-se o banco de dados eXist para o armazenamento das ontologias, evitando que as mesmas

estejam dispostas em arquivos em uma estrutura de diretórios. O eXist permite que sejam processadas diversas ontologias ao mesmo tempo, facilitando a integração de ontologias que descrevem domínios similares.

7. Consultas à ontologia com SPARQL

Com base em um grupo de onze tarefas pré-definidas e divisíveis em subgrupos com a finalidade de gerar novas tarefas (apresentada na seção 4), foi construída uma ontologia, em OWL DL, capaz de processar esta informação e relacionar tarefas entre clínicos e pacientes. Foi usada a ferramenta Protégé que, através de uma interface gráfica, permite a construção de uma ontologia. Além desta ferramenta, o sistema Ontohealth também foi utilizada, para processar as informações presentes nesta ontologia, através de funções próprias e consultas em SPARQL.

SPARQL, que é um protocolo e uma linguagem de consulta sobre RDF [Clark 2005]. Assim como o RDF, a SPARQL também é construída com triplas padrão e é constituída de um sujeito, um predicado e um objeto.

A ontologia construída para esta aplicação foi estruturada de uma forma simples, possuindo apenas 2 classes principais: *Clínico* e *Paciente*. A classe *Clínico* é a responsável por realizar um conjunto de onze tarefas já definidas, além de poder criar novas tarefas a partir destas já existentes. Essas novas tarefas que podem ser criadas por um clínico, são subclasses de novas tarefas, e possuem como propriedades um subconjunto das 11 tarefas já definidas. É através dessas propriedades que é possível fazer relações entre clínicos e pacientes. Com a utilização desta ontologia em um ambiente pervasivo, médicos poderão ter acesso a histórico clínico de pacientes, definir tarefas tendo um controle sobre o que é feito, quando é feito, onde é feito e por quem é feito, e saber se há alguma restrição que impeça a execução de determinado conjunto de tarefas.

Antes de mostrar consultas SPARQL baseadas na ontologia desenvolvida para um ambiente hospitalar, é necessário analisar uma parte desta ontologia, onde são declarados os indivíduos, como mostra o código OWL a seguir:

```
<Paciente rdf:ID="Maria_Lucia">
   <Idade rdf:datatype="http://www.w3.org/2001/XMLSchema#int">60</Idade>
   <Nome rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Maria Lucia</Nome>
   <Codigo rdf:datatype="http://www.w3.org/2001/XMLSchema#int">10</Codigo>
</Paciente>

<Paciente rdf:ID="Pedro">
   <Idade rdf:datatype="http://www.w3.org/2001/XMLSchema#int">49</Idade>
   <Nome rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Pedro</Nome>
   <Codigo rdf:datatype="http://www.w3.org/2001/XMLSchema#int">20</Codigo>
</Paciente>

<Paciente rdf:ID="Joao_Carlos">
   <Nome rdf:datatype="http://www.w3.org/2001/XMLSchema#string">João Carlos</Nome>
</Paciente>

<Clinico rdf:ID="Mario">
   <Nome rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mario</Nome>
   <Codigo rdf:datatype="http://www.w3.org/2001/XMLSchema#int">111</Codigo>
   <Idade rdf:datatype="http://www.w3.org/2001/XMLSchema#int">48</Idade>
</Clinico>
```

Para buscar todos os indivíduos que atualmente fazem parte do ambiente hospitalar, deve-se realizar uma consulta como a seguinte:

```
SELECT ?codigo ?nome ?idade
WHERE {
   ?var :Nome ?nome.
   OPTIONAL {
      ?var :Codigo ?codigo.
      ?var :Idade ?idade.
   }
}
```

Observe que a cláusula *Optional* está sendo utilizada. Isto é necessário, pois de acordo com o código OWL descrito acima, um dos indivíduos não possui um código e idade cadastrados, portanto ele seria omitido do resultado. Se esta cláusula não fosse utilizada, o resultado traria apenas os indivíduos que tem todos os seus dados cadastrados. Na figura 7, é possível perceber que, mesmo com alguns campos em branco, o indivíduo *João Carlos* também aparece no resultado da pesquisa.

row	codigo	nome	idade
1		João Carlos	
2	10	Maria Lucia	60
3	20	Pedro	49
4	111	Mario	48

Figura 7. Uso do recurso *Optional*

A linguagem SPARQL possui outro recurso interessante chamado *Filter* que possibilita a restrição do conjunto de soluções de acordo com a definição de expressões. As expressões podem ser funções e operações, sendo que os operandos dessas funções e operadores são um subconjunto dos tipos de dados do XML Schema (*xsd:string*, *xsd:decimal*, *xsd:double*, *xsd:dateTime*) e tipos derivados de *xsd:decimal* [Duckett et al. 2001].

O código a seguir mostra uma consulta que busca os dados de todos os pacientes que tem idade entre 45 e 50 anos.

```
SELECT ?codigo ?nome ?idade
WHERE {
   ?var :Nome ?nome.
   ?var :Idade ?idade.
   OPTIONAL {
      ?var :Codigo ?codigo.
   }
   FILTER (?idade>=45 && ?idade<=50)
}
```

Percebe-se que, a variável *?idade* não está dentro da cláusula *Optional* pois neste caso é necessário saber a idade dos indivíduos para chegar à resposta. O resultado desta pesquisa é mostrado na figura 8.

row	codigo	nome	idade
1	20	Pedro	49
2	111	Mario	48

Figura 8. Uso do recurso *Filter*

Além disso, a linguagem SPARQL ainda possui recurso de ordenação (*Order by*), recurso para limitar o número de resultados (*Limit*) e também limitar o número de resultados por página (*Offset*). Um exemplo disto é mostrado a seguir:

```
SELECT ?codigo ?nome ?idade
WHERE {
   ?var :Nome ?nome.
   ?var :Idade ?idade.
   OPTIONAL {
      ?var :Codigo ?codigo.
   }
}
ORDER BY DESC(?idade)
LIMIT 4
OFFSET 2
```

Pela análise do código pode-se dizer que o resultado desta consulta trará todos os indivíduos que tem os campos nome e idade cadastrados ordenados do mais velho ao mais jovem. Além disso, a cláusula Limit diz que somente os quatro pacientes mais velhos serão listados. E por último, a cláusula *Offset* limita a dois resultados mostrados em cada página. Confira o resultado nas figuras 9 e 10, sendo que cada uma representa uma página.

row	codigo	nome	idade
1	10	Maria Lucia	60
2	20	Pedro	49

Figura 9. Resultado da pesquisa - página 1

row	codigo	nome	idade
3	111	Mario	48

Figura 10. Resultado da pesquisa - página 2

Percebe-se que mesmo com o limite sendo igual a quatro, apenas três resultados foram mostrados, pois existiam somente três indivíduos com os campos nome e idade cadastrados.

8. Trabalhos Relacionados

Analisando as idéias propostas neste artigo, é possível relaciona-lo com outros projetos com foco no uso de ontologias em ambientes pervasivos. O projeto CoBrA [Chen and Finin 2003], assim como este projeto, utiliza uma ontologia própria, chamada CoBrA-Ont, desenvolvida em OWL. O CoBrA foi desenvolvido para ser utilizado em ambientes inteligentes, tendo estruturas físicas e lógicas específicas. O projeto apresentado neste artigo terá como ambiente inteligente um ambiente médico-clínico, como por exemplo, um hospital.

O Jena[1] é um *framework* para a construção de aplicações para a Web Semântica. Ele fornece um ambiente para a manipulação de ontologias em linguagens como OWL. Contudo, o Jena tem alguns problemas no que diz sentido às consultas sobre a ontologia, utilizando a linguagem SQWRL. Até o momento, não há uma integração entre esta linguagem de consulta e o Jena. Desta forma, optou-se pela construção de um sistema específico para a manipulação de ontologias hospitalares, que aborde não só a linguagem SQWRL, mas também as linguagens XQuery e XPath.

[1] http://jena.sourceforge.net/

Por sua vez, o *Context Management System* (CxMS) [Jahnke et al. 2005] tem como objetivo a implantação de um sistema sensível ao contexto [Schilit and Theimer 1994] em um ambiente hospitalar. Foram usadas ontologias base, tendo como as mais importantes a *HL7 Reference Information Model* (RIM) e a *HL7 Clinical Document Architecture* (CDA). Utilizando a ferramenta Microsoft SharePoint que serve para criação de websites sensíveis a contexto em uma Intranet, com suporte a documentos Office [Jahnke et al. 2005].

9. Conclusão

Acredita-se que, no futuro, os sistemas de informação hospitalares poderão reter conhecimento sobre as melhores práticas clínicas, financeiras e operacionais de diagnóstico e tratamento de patologias, para que o prolongamento de vida com qualidade de pacientes seja sempre a principal missão dos hospitais.

A Computação Pervasiva se mostra uma etapa indispensável a ser consolidada no caminho de propostas como a de Weiser. Mesmo tendo uma premissa mais próxima das tecnologias de *hardware* e *software* atualmente praticadas, a Computação Pervasiva constituirá ainda um campo fértil para ofertas de produtos e desenvolvimento de pesquisas nos próximos anos [Saha and Mukherjee 2003].

Assim, tem-se a idéia de que um hospital seja visto como um ambiente pervasivo, de modo que os componentes da aplicação e serviços possam usufruir da mobilidade dentro deste ambiente. Com a ontologia proposta no artigo e a ferramenta para seu processamento, as tarefas do dia-a-dia hospitalar poderão ser compartilhadas por todos os componentes deste ambiente pervasivo.

O sistema Ontohealth aparece como uma forma eficiente de realizar o processamento da ontologia que representa um ambiente hospitalar pervasivo. Esse sistema fornece uma série de programas (métodos) desenvolvidos para auxiliar médicos na criação de tarefas e fluxos de trabalho para serem executados e assim atender seus pacientes de forma mais ágil e confiável, sempre visando o bem-estar dos mesmos.

O próximo passo neste projeto é integrar essa ontologia com algum sistema EHR, como o UMLSKS [Bangalore et al. 2003]. O *UMLS Knowledge Source Server* (UMLSKS) é um sistema para auxiliar profissionais e pesquisadores da saúde e integrar sistemas de informação biomédicos a partir de uma variedade de banco de dados bibliográficos e sistemas especialistas. Utilizar-se-á de ferramentas, como o Metamorphosis [Librelotto et al. 2006], para a geração automática de ontologias em OWL a partir dos arquivos do UMLSKS. Com isso, a ontologia abrangerá um domínio ainda mais amplo e padronizada pelos termos do UMLSKS.

Como outras formas de dar seguimento a este trabalho, está previsto a especificação de novos métodos para o sistema OntoHealth, visando oferecer um auxilio cada vez maior aos médicos. Planeja-se, também, desenvolver um motor de inferência, onde pode ser definindo conjuntos de regras e estratégias de busca visando aumentar o desempenho das pesquisas realizadas com o OntoHealth. Por fim pretende-se realizar testes em um ambiente hospitalar pervasivo real.

Referências

Bangalore, A., Thorn, K. E., Tilley, C., and Peters, L. (2003). The UMLS Knowledge Source Server: An Object Model for Delivering UMLS Data. AMIA Annual Symposium Proceedings.

Chandrasekaran, B. (1999). What Are Ontologies, and Why do We Need Them? In *IEEE Inteligent Systems and their applications*, volume vl 9, n 1. IEEE.

Chen, H. and Finin, T. (2003). An ontology for a context aware pervasive computing environment. In *IJCAI workshop on ontologies and distributed systems*. Acapulco MX.

Clark, K. (2005). SPARQL: Web 2.0 Meet the Semantic Web. http://www.oreillynet.com/xml/blog/2005/09/sparql_web_20_meet_the_semanti.html.

Duckett, J., Griffin, O., Mohr, S., Norton, F., Ozu, N., Stokes-Rees, I., Tennison, J., Williams, K., and Cagle, K. (2001). *Professional XML Schemas*. Wrox Press.

Fensel, D., van Harmelen, F., Horrocks, I., McGuinness, D., and Patel-Schneider, P. F. (2001). Oil: An ontology infrstructure for the semantic web. In *IEEE Intelligent Systems*, volume 16(2), pages 38–44. IEEE.

Ferreira, G. L., Librelotto, I. A. G. R., Silva, F. L., and Yamin, A. (2009). Middleware for management of end-user programming of clinical activities in a pervasive environment. In *Workshop on Middleware for Ubiquitous and Pervasive Systems*, volume 389, pages 07–12.

Guarino, N. (1997). Semantic Matching: Formal Ontological Distinctions for Information Organization, Extraction, and Integration. In *Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology*, pages 139–170. Springer Verlag.

Horrocks, I. (2008). Ontologies and the semantic web. In *Communications of the ACM*, volume 12. ACM (The Association for Computing Machinery).

Jahnke, J. H., Bychkov, Y., Dahlem, D., and Kawasme, L. (2005). Context-aware information services for health. Department of Computer Science - University of Victoria.

Kumar, A., Ciccarese, P., Smith, B., and Piazza, M. (2003). Context-based task ontologies for clinical guidelines. Ontologies in Medicine: Proceedings of the Workshop on Medical Ontologies, IOS Press.

Laerum, H. and Faxvaag, A. (2004). Task-oriented evalution of eletronic medical record systems: development and validation of a questionaire for physicians. In *BMC Medical Informatics and Decision Making 2004*, volume vl 4, n 1.

Librelotto, G. R., Gassen, J. B., Freitas, L. O., da Silva, F. L., Augustin, I., and Henriques, P. R. (2008). Uma Ontologia aplicada a um Ambiente Pervasivo Hospitalar. In *VIII Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI 2008)*, page 34.

Librelotto, G. R., Gassen, J. B., Freitas, L. O., Vizzotto, J. K., da Silva, F. L., and Augustin, I. (2009). A Definição de uma API para o Processamento de Ontologias em Hospitais Pervasivos. In *V Simpósio Brasileiro de Sistemas de Informação (SBSI 2009)*, pages 01–11.

Librelotto, G. R., Ramalho, J. C., and Henriques, P. R. (2006). Metamorphosis - A Topic Maps Based Environment to Handle Heterogeneous Information Resources. In *Lecture Notes in Computer Science*, volume 3873, pages 14–25. Springer-Verlag GmbH.

Mcguineness, D. L. and Harmelen, F. V. (2004). OWL - Web Ontology Language overview. http://www.w3.org/TR/owl-features.

Meier, W. (2008). eXist - Open Source Native XML Database. http://www.exist-db.org.

Saha, D. and Mukherjee, A. (2003). Pervasive Computing: a Paradigm for th 21st Century. In *IEEE Computer, New York*, volume 36, pages 25–31. IEEE Computer Society.

Schilit, B. and Theimer, M. (1994). Disseminating active map information to mobile hosts. IEEE Network.

Swartout, W. and Tate, A. (1999). Ontologies. In *IEEE Inteligent Systems and their applications*, volume vl 14, n 1. IEEE.