Descoberta Semântica de Recursos na Computação Ubíqua

Renato M. Dilli[1], Sérgio L. Rodrigues[1], Nelsi Warken[2], Luthiano R. Venecian[3], João L. B. Lopes[1,5], Adenauer C. Yamin[3,4], Claudio F. R. Geyer[5]

[1]Instituto Federal Sul-rio-grandense (IFSul)

[2]Empresa Brasileira de Pesquisa Agropecuária (Embrapa)

[3]Universidade Católica de Pelotas (UCPel)

[4]Universidade Federal de Pelotas (UFPel)

[5]Universidade Federal do Rio Grande do Sul (UFRGS)

{dilli,sergio,jlopes_cavg}@ifsul.edu.br, nelsi.warken@cpact.embrapa.br

{venecian,adenauer}@ucpel.tche.br, geyer@inf.ufrgs.br

Abstract. *In ubiquitous environments, the resources must be shared so that they can be accessed anywhere and anytime. In this sense, the process of resource discovery plays an important role. This paper presents the EXEHDA-SD (EXEHDA-Semantic Discovery), a mechanism for resource discovery in ubiquitous computing. This mechanism combines in its architecture technologies for semantic processing of resources requests. This way, we intend to increase the expressiveness of the resource representation and query. The mechanism also provides the dynamics with which resources enter and leave the environment and include aspects such as scalability and user preferences.*

Resumo. *Em ambientes ubíquos, os recursos devem estar compartilhados para que possam ser acessados de qualquer lugar e a qualquer momento. Nesse sentido, o processo de descoberta de recursos assume um importante papel. Este artigo apresenta o EXEHDA-SD (EXEHDA-Semantic Discovery), um mecanismo para descoberta de recursos na computação ubíqua, que agrega em sua arquitetura tecnologias para o processamento semântico de requisições por recursos. Com isso, busca-se aumentar a expressividade na representação e consulta de recursos. O mecanismo também prevê a dinamicidade com que os recursos entram e saem do ambiente e contempla aspectos como escalabilidade e preferências do usuário.*

1. Introdução

Os ambientes ubíquos são compostos por uma grande quantidade de recursos heterogêneos, dispersos e de disponibilidade variável [Zhu et al. 2005].

Um mecanismo para descoberta de recursos deve considerar a dinamicidade com que os recursos entram e saem do ambiente computacional. Estes mecanismos devem gerenciar a relação entre os consumidores e provedores de recursos, permitindo a localização dos recursos disponíveis, com pouca ou nenhuma intervenção do usuário [Robinson and Indulska 2007].

O emprego de tecnologias semânticas eleva a expressividade na representação e consulta dos recursos, bem como permite a definição de regras para consistência e extensão das consultas. O uso de ontologias na representação dos recursos torna a con-

sulta mais precisa, evitando ambiguidade de conceitos e possibilitando a realização de inferência sobre o modelo ontológico [Soldatos et al. 2007].

Considerando este cenário, propomos o EXEHDA-*Semantic Discovery* (EXEHDA-SD), um mecanismo para descoberta de recursos com suporte semântico que visa qualificar a descoberta de recursos do *middleware* EXEHDA [Lopes et al. 2007] no ambiente ubíquo gerenciado pelo mesmo. Este ambiente é formado por células compostas por: (i) um EXEHDAbase, responsável por todos os serviços básicos da célula; (ii) EXEHDAnodos, são os equipamentos de processamento disponíveis no ambiente; (iii) EXEHDAmob-nodos, são os nós do sistema com elevada portabilidade.

O modelo proposto também busca suprir as demandas inerentes aos ambientes ubíquos, dentre as quais destacam-se: escalabilidade, preferências do usuário, notificação do cliente quando o recurso desejado estiver disponível, controle de acesso a recursos de acordo com o perfil do usuário e controle da disponibilidade de recursos através de intervalos de tempo [Costa et al. 2008].

Esse trabalho desenvolve-se no âmbito do grupo de pesquisa G3PD do PPGInf/UCPel, tendo sido um *position paper* com o andamento do mesmo apresentado no SBCUP 2010. O presente artigo traduz os resultados completos da pesquisa e está organizado nas seguintes seções: a seção 2 apresenta o modelo arquitetural baseado em processamento semântico; a seção 3 destaca um estudo de caso desenvolvido na Empresa Brasileira de Pesquisa Agropecuária (Embrapa); a seção 4 sistematiza e compara os trabalhos relacionados e a seção 5 apresenta as considerações finais.

2. Modelo Arquitetural

A arquitetura de software do EXEHDA-SD, apresentada na Figura 1, está organizada em três componentes distintos: Componente Cliente (CC), Componente Recurso (CR) e Componente Diretório (CD).

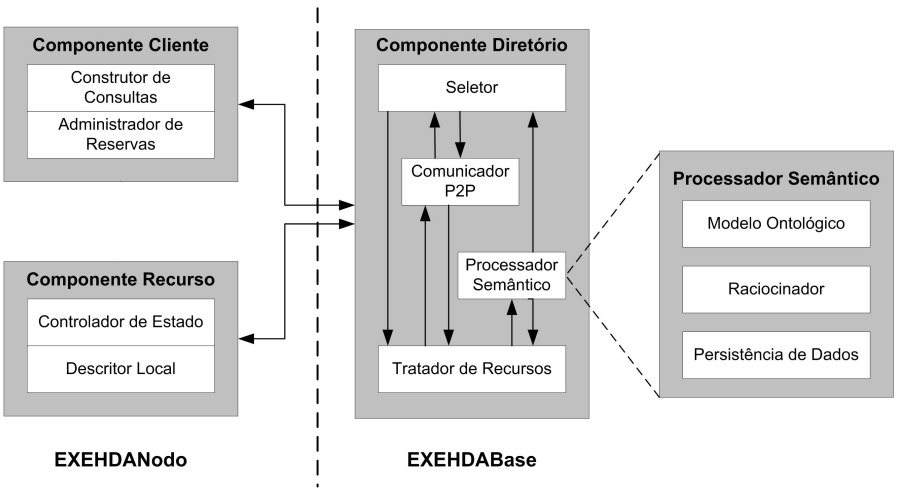

Figura 1. Modelo Arquitetural

2.1. Componente Cliente

O CC é responsável pela especificação da consulta aos recursos desejados, definindo os parâmetros a serem empregados no processamento da mesma. Dependendo da natureza

da consulta podem ser utilizadas as preferências do cliente, caracterizando assim, um mecanismo organizado em dois níveis. Os clientes são os componentes de software das aplicações em execução.

O CC é composto pelos seguintes módulos: (i) Construtor de Consultas, responsável por processar os requisitos para descoberta de recursos da aplicação, e gerar o arquivo XML com a especificação da correspondente pesquisa, bem como receber as respostas das consultas realizadas; (ii) Administrador de Reservas, responsável por notificar o cliente quando o diretório verificar que o recurso desejado tornou-se disponível.

2.2. Componente Recurso

O CR é responsável por notificar o estado atual recurso. Isso é feito através de troca de mensagens com o CD, considerando o intervalo de tempo especificado para o recurso (*lease*). Outra função do CR é anunciar os recursos dos nodos a medida que estes se deslocam no ambiente ubíquo. Quando um nodo ingressa em uma nova célula, o CR envia o arquivo OWL [McGuinness and van Harmelen 2009] com a descrição dos seus recursos para o CD da célula que está sendo visitada. O CD, por sua vez, adiciona as informações descritas em OWL na ontologia da célula que está sendo visitada. O CR é composto pelos módulos Controlador de Estado e Descritor local.

O Controlador de Estado tem por finalidade anunciar a disponibilidade do recurso no ambiente ubíquo. O anúncio é feito enviando uma mensagem ao Tratador de Recursos do Diretório. A técnica utilizada pelo mecanismo para controlar o estado dos recursos é *soft state*. Como o ambiente ubíquo pode conter uma grande quantidade de recursos, a melhor alternativa é os recursos se anunciarem para o CD. As mensagens de anúncio ocorrem após o CD confirmar o registro do recurso no diretório da célula ([Dilli 2010]).

O Descritor Local tem por objetivo armazenar e manter atualizado o arquivo OWL com as descrições do recurso localmente. Este arquivo é gerado pela interface de cadastro de recursos, localizada no diretório, no momento em que o recurso é instanciado na ontologia, ou são alterados atributos. Este módulo recebe um novo arquivo de descrições em OWL sempre que o recurso sofrer alterações no descritor, localizado no diretório. A cada ativação do recurso na célula, o Descritor Local envia o arquivo de descrições em OWL para o Tratador de Recursos.

2.3. Componente Diretório

O CD está localizado no EXEHDABase de cada célula do ambiente ubíquo. Esse componente é formado por quatro módulos descritos a seguir.

2.3.1. Tratador de Recursos

Quando um CC solicita que seja notificado sobre a disponibilidade de algum recurso é realizada uma consulta sobre todos os recursos “Ativos” no ambiente ubíquo. Caso a consulta não retorne resultados que atendam à requisição, a consulta será refeita fazendo uso dos recursos “Inativos”, mas existentes na ontologia. A consulta retornando recursos que satisfaçam a requisição, o Tratador de Recursos notifica o CC quando o CR do recurso desejado renovar o *lease*, alterando o estado do recurso para “Ativo”. No estado

"Ativo" o recurso estará disponível para ser alocado pelo *Resource Broker* do *middleware* EXEHDA.

O Controle de Recursos Ativos (CRA) é um repositório, mantido pelo Tratador de Recursos, onde são armazenados o identificador do nodo (Nodo_ID) e o intervalo de tempo gerenciado pelo Tratador de Recursos (*lease*). O CRA gerencia os recursos que entram e saem do ambiente celular. Quando um recurso entra no ambiente o CR envia uma mensagem para o Tratador de Recursos e este adiciona o Nodo_ID onde está localizado o recurso e o *lease* padrão definido pelo mecanismo. O CR precisa renovar o *lease* do recurso localizado no CD periodicamente, caso contrário o Tratador de Recursos irá remover o recurso do CRA.

Quando uma pesquisa é recebida pelo Tratador de Recursos, e no seu perfil está declarada a solicitação de notificação de recursos disponíveis, este módulo armazena no repositório Controle de Notificação de Recursos Disponíveis (CNR) (i) o Nodo_ID do recurso desejado, mas que não está disponível no momento; (ii) o ID do Cliente que solicitou a pesquisa; (iii) a consulta; (iv) a data e a hora que foi feita a pesquisa; (v) a data e a hora que irá expirar a espera pelo recurso desejado.

Quando o CR anuncia um recurso ao Tratador de Recursos, este verifica se o recurso está no CRA e renova seu *lease* ou adiciona o recurso com o *lease* padrão caso não esteja presente do CRA. Após, o Tratador de Recursos consulta o CNR para verificar se o recurso não está sendo esperado por algum cliente. Caso positivo, o cliente será notificado que o recurso está disponível.

Este módulo também tem por objetivo controlar o agendamento da disponibilidade do recurso, especificando, por exemplo, dias da semana e horários em que o recurso está disponível no ambiente.

2.3.2. Processador Semântico

O Processador Semântico emprega a API Jena [Dickinson 2009], sendo formado pelos seguintes componentes: (i) Modelo Ontológico (vide Figura 2), responsável pela manutenção da ontologia do mecanismo de descoberta. Esta ontologia é desenvolvida com a linguagem OWL. (ii) Raciocinador, responsável por aplicar regras e processar as consultas realizadas em SPARQL [Prud'hommeaux and Seaborne 2008]. O raciocinador procura por recursos idênticos e semelhantes ao solicitado; (iii) Persistência de Dados, responsável pelo armazenamento das triplas da ontologia no banco de dados PostgreSQL.

O modelo ontológico utilizado pelo Processador Semântico está fundamentado em duas ontologias: (i) OntUbi: desenvolvida pelo grupo G3PD da Universidade Católica de Pelotas para ser utilizada pelos mecanismos de consciência e adaptação de contexto e descoberta de recursos; (ii) OntSD: desenvolvida para atender exclusivamente o EXEHDA-SD. Esta ontologia integra-se à OntUbi através de vários relacionamentos entre classes.

O Processador Semântico é responsável pela instanciação de novos recursos e processamento de regras de inferência, possibilitando ao mecanismo de descoberta o reconhecimento de conceitos implícitos na ontologia. Para processar as regras é executado um código em Java que adiciona os conceitos deduzidos pelas regras na ontologia e realiza a consulta através da linguagem SPARQL, considerando o novo conceito. Através

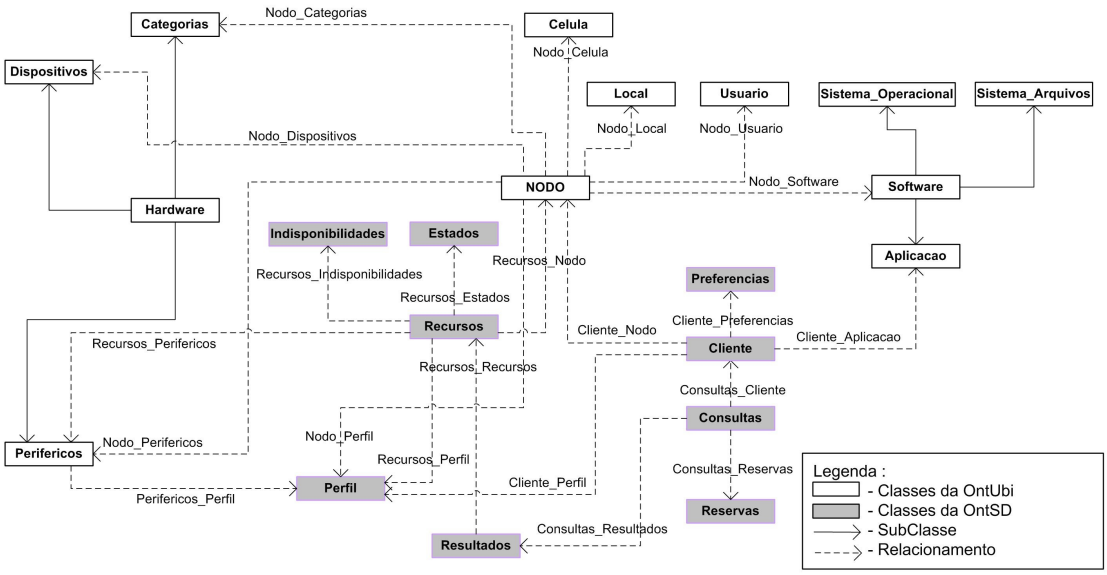

Figura 2. Modelo Ontológico

do suporte à persistência de dados da API Jena, as triplas da ontologia são armazenadas, tornando possível executar consultas SPARQL diretamente no banco de dados.

2.3.3. Comunicador P2P

O Comunicador P2P é o módulo responsável pela comunicação com as células vizinhas. A comunicação entre as células é realizada através de mecanismo P2P entre os EXEHDABase de cada célula. Utiliza-se uma variação do P2P puro, chamada *super peer* ([Dilli 2010]). Neste modelo a comunicação cliente/servidor ocorre apenas entre os diretórios localizados nos EXEHDABase de cada célula. Os CC e CR acessam apenas o diretório da célula local. O Comunicador P2P é responsável por repassar a pesquisa para as células vizinhas de acordo com o número de saltos definidos pelo CC.

O fluxo de informação entre os Comunicadores P2P está organizado em duas fases: (i) Configuração Celular: instanciação e presença de recursos no ambiente celular; (ii) Pesquisa por Recursos: nesta fase o fluxo de informação acontece a partir da consulta de recursos entre células, originada por um cliente. Essas fases estão organizadas em 11 etapas, conforme os itens a seguir. A Configuração Celular corresponde às etapas 1 a 3 e a Pesquisa por Recursos diz respeito às etapas 4 a 11.

1. Administrador do ambiente instancia um novo recurso no modelo ontológico e o Processador Semântico envia o arquivo no formato OWL com as descrições do recurso cadastrado ao Tratador de Recursos;
2. Tratador de Recursos envia o arquivo OWL para o módulo Descritor Local do CR que contém os recursos instanciados no CD;
3. Controlador de estado do CR envia uma mensagem ao Tratador de Recursos para anunciar sua presença no ambiente, tornado-se disponível;
4. Módulo Construtor de Consultas gera um arquivo XML contendo as características do recurso desejado pelo cliente e envia para o Tratador de Recursos, no CD

de sua célula;
5. Tratador de Recursos realiza a leitura do arquivo XML e identifica que o escopo de pesquisa envolve várias células e repassa este XML para o Comunicador P2P. O Tratador de recursos interpreta a consulta especificada em XML, gerando uma nova consulta na linguagem SPARQL e repassa ao Processador Semântico juntamente com as regras de inferência especificadas no arquivo XML, se existirem;
6. Processador Semântico processa a consulta e as regras de inferência no modelo ontológico da célula local e repassa os resultados da consulta para o Seletor. O Comunicador P2P envia o arquivo XML para células vizinhas utilizando tecnologia *super peer*. Também é verificado se o escopo de pesquisa envolve células estáticas definidas pelo administrador do ambiente;
7. Comunicador P2P de uma célula recebe o arquivo XML e a mensagem do P2P cliente contendo o ID da Base que originou a consulta. O arquivo XML é repassado para o Tratador de Recursos;
8. Tratador de Recursos interpreta o arquivo XML e gera a consulta na linguagem SPARQL e repassa ao Processador Semântico com regras de inferência, se especificadas;
9. Processador Semântico processa a consulta no modelo ontológico e repassa os resultados ao Seletor;
10. Seletor aplica as preferências do usuário, se especificadas, e entrega os resultados da célula ao Comunicador P2P;
11. Comunicador P2P decrementa a profundidade de pesquisa informada na mensagem recebida pelo P2P origem e repassa os resultados da pesquisa ao Comunicador P2P que solicitou a pesquisa.

2.3.4. Seletor

Quando uma pesquisa retorna mais de um recurso, o módulo Seletor faz a classificação e ordena os recursos, posicionando os que melhor satisfazem a requisição no topo da lista. O Seletor é responsável por receber e organizar as respostas das consultas realizadas em outras células, recebidas pelo módulo P2P. O processamento do resultado da consulta envolvendo várias células coloca no topo da lista os recursos que estiverem: (i) localizados na própria célula que originou a consulta; (ii) nas células vizinhas estáticas; (iii) em células com menor profundidade de pesquisa.

As preferências do cliente definidas no arquivo XML do Componente Cliente são consideradas pelo Seletor na composição do resultado da pesquisa.

3. Estudo de Caso

As funcionalidades do EXEHDA-SD tem sido avaliadas através de estudos de caso desenvolvidos na Embrapa Clima Temperado em Pelotas, RS. O estudo de caso apresentado nesse artigo explora os aspectos de escalabilidade, preferências do cliente, perfil de acesso e expressividade na representação e consulta por recursos através do Processador Semântico. Para a execução do cenário proposto a estrutura física da Embrapa é organizada em três células, conforme Figura 3, as quais são instanciadas no modelo ontológico, através do Tratador de Recursos.

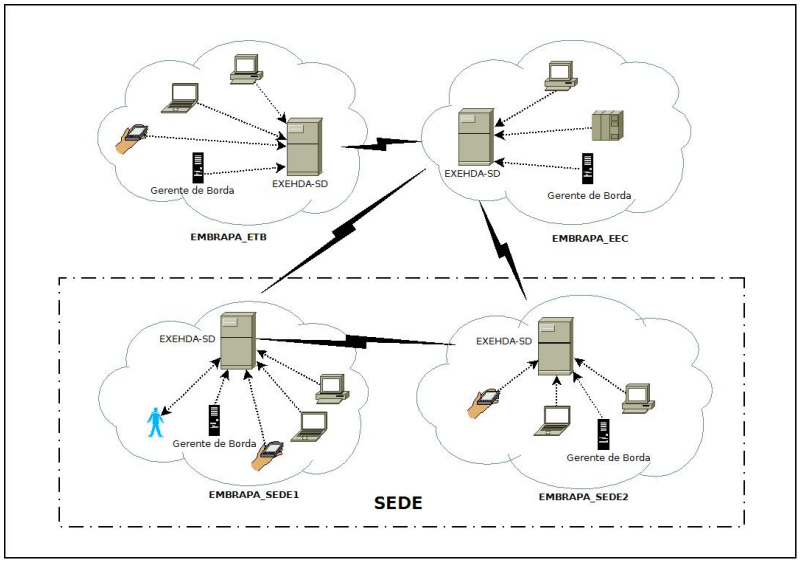

Figura 3. Organização Celular

No cenário proposto são gerenciados os recursos envolvidos nas casas de vegetação, as quais constituem ambientes monitorados e controlados para realização de experimentos pelos pesquisadores. O espaço disponível para utilização pelo pesquisador dentro da casa de vegetação é chamado de *slot*. Cada casa de vegetação possui um número limitado de *slots* que são considerados recursos a serem gerenciados pelo EXEHDA-SD. Os recursos gerenciados são instanciados no modelo ontológico através do administrador do ambiente com a interface disponibilizada pelo Tratador de Recursos.

Nesse cenário, um pesquisador localizado na célula EMBRAPA_SEDE1 executa uma aplicação, localizada em seu nodo, que tem por objetivo encontrar *slots* livres, em casas de vegetação, que possuam sensor e atuador de umidade. A regra de inferência construída pelo cliente cria um novo conceito no modelo ontológico relacionado a canteiros de vegetação que possuam pelo menos um atuador e um sensor de umidade. A regra é instanciada no modelo ontológico, conforme a Figura 4. A aplicação permite selecionar o perfil do cliente, a célula em que deseja pesquisar, o conceito deduzido (criado através de regras de inferência), a situação do *slot* no canteiro de vegetação (livre ou ocupado), local do canteiro, nome do canteiro, tipo de atuador e tipo de sensor.

Para processar a solicitação do pesquisador, o Componente Cliente gera o XML com as especificações de recursos necessárias para realizar a consulta e envia ao Tratador de Recursos do Componente Diretório da célula (informações adicionais sobre a forma de especificação dos recursos são encontradas em [Dilli 2010]). Por sua vez, o Processador Semântico traduz e processa a consulta de XML para SPARQL, envia para o Seletor aplicar as preferências do usuário. Por fim, o Tratador de Recursos envia no formato XML o resultado ao CC.

A Figura 5 mostra o resultado obtido pelo cliente. Os slots 3, 6 e 45 possuem sensores e atuadores de umidade. Os dois primeiros estão localizados na mesma célula (celula_10) em que o cliente estava no momento da solicitação da consulta. O slot 45, localizado na "celula_6", é exibido em função da profundidade de pesquisa definida pelo cliente (vide Figura 4), a qual altera o escopo da consulta para utilizar células vizinhas

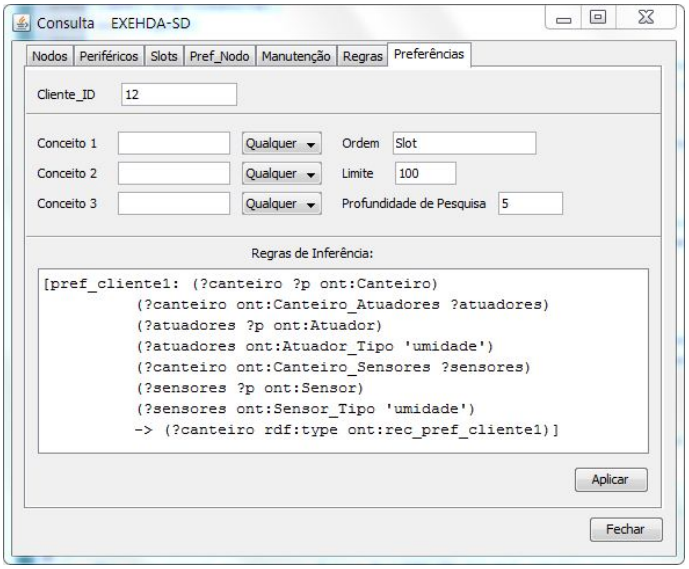

Figura 4. Preferências do Cliente

estáticas e dinâmicas. Como está definido o valor "5" na profundidade de pesquisa, os Comunicadores P2P replicam a consulta aos Comunicadores P2P de células vizinhas, até o quinto nível.

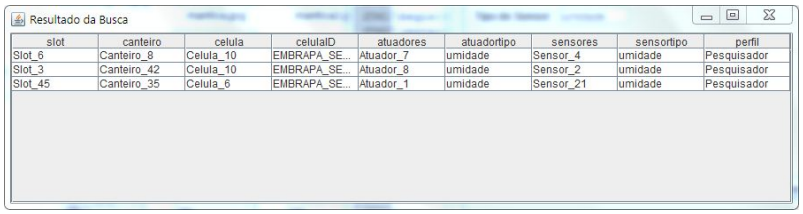

Figura 5. Resultado da Consulta

O EXEHDA-SD possibilita localizar recursos dispersos entre as células do ambiente ubíquo de forma escalável. Desta maneira o ambiente pode expandir-se de forma dinâmica, enquanto as aplicações executam, sem comprometer aspectos funcionais do mecanismo de descoberta.

4. Trabalhos Relacionados

[Allemand 2006] utiliza ontologias descritas em OWL para representação dos recursos e a linguagem RDQL (*RDF Data Query Language*) para consultar recursos instanciados no modelo ontológico. O projeto foi concebido para uso em ambientes de grade computacional. Esse projeto não considera as preferências do usuário, ao contrário do EXEHDA-SD. O EXEHDA-SD também diferencia-se pelo uso da linguagem de consulta SPARQL, considerada uma evolução da RDQL, sendo recomendada pelo W3C como padrão para realização de consultas em ontologias descritas com OWL.

DReggie [Chakraborty et al. 2001] consiste em um modelo para atuar com aplicações de comércio eletrônico em dispositivos móveis. Para isso é utilizada a linguagem DAML na descrição de recursos e um raciocinador Prolog para realizar *matching*

na base de conhecimento. O EXEHDA-SD utiliza raciocinadores disponíveis na API Jena e foi modelado para atuar tanto com dispositivos fixos como móveis, espalhados pelo ambiente ubíquo.

OMM [Tangmunarunkit et al. 2003] é um serviço de *matchmaking* em ambiente de grade. Este projeto utiliza tecnologias de Web Semântica para o *matching* de recursos no ambiente distribuído. As consultas são realizadas através da linguagem TRIPLE sobre ontologias descritas em RDF *Schema*. O EXEHDA-SD utiliza OWL para representação das ontologias. Essa linguagem é recomendada pela W3C e se traduz em uma evolução de outras linguagens, como RDF *Schema* e DAML. De modo diferente deste projeto e dos já referidos, o EXEHDA-SD tem uma maior abrangência de descoberta, com um escopo que envolve ambientes ubíquos e não somente ambientes de grade ou *mobile commerce*.

PerDiS [Schaeffer 2005] foi modelado para descoberta de recursos em ambientes ubíquos. Assim como o EXEHDA-SD, utiliza redes *super-peer* para localização de recursos entre células. Um diferencial entre o EXEHDA-SD e o PerDiS está na representação e consulta por recursos. No EXEHDA-SD isso é feito através de processamento semântico, empregando ontologia descrita com OWL e consulta com SPARQL. Já no PerDiS os recursos são representados em XML e a consulta é realizada pela comparação de atributos e valores, utilizando XML. Assim, ao contrário do EXEHDA-SD, o PerDiS permite apenas *matching* sintático e não possui motores de inferência para dedução de recursos.

A Tabela 1 apresenta uma comparação entre os trabalhos relacionados, incluindo o EXEHDA-SD. As seguintes características são consideradas na comparação: (i) a linguagem utilizada para descrição e consulta por recursos; (ii) a abrangência de descoberta do mecanismo; (iii) o emprego de algum tipo de preferência do usuário para a realização da consulta ou processamento de resultados.

Tabela 1. Comparação entre os Trabalhos Relacionados e o EXEHDA-SD

Modelos	Descrição	Consulta	Abrangência	Preferências
Allemand	OWL	RDQL	Grade	-
DReggie	DAML	Prolog	*m-commerce*	Sim
OMM	RDF *Schema*	TRIPLE	Grade	Sim
PerDiS	XML	XML	Ubíquo	Sim
EXEHDA-SD	OWL	SPARQL	Ubíquo	Sim

Além das diferenças destacadas nos parágrafos anteriores, o EXEHDA-SD contempla as seguintes características que não foram identificadas nos trabalhos relacionados: (i) perfil definido nos recursos e clientes para realizar o *matching*; (ii) notificação ao cliente quando o recurso desejado tornar-se disponível no ambiente ubíquo; (iii) agendamento de indisponibilidades para definir períodos de tempo em que o recurso não estará acessível no ambiente.

5. Considerações Finais

O principal diferencial do EXEHDA-SD em relação aos trabalhos relacionados diz respeito a maior expressividade na representação e consulta de recursos. Isso é obtido através da concepção de um modelo arquitetural baseado em ontologias, que provê processamento semântico das requisições por recursos.

Na continuidade da pesquisa do EXEHDA-SD os seguintes aspectos poderão ser explorados em trabalhos futuros: (i) contemplar a interoperabilidade de ontologias, possibilitando que cada célula possua um modelo ontológico diferenciado, mas equivalente; (ii) agregar novos algoritmos no processo de *ranking* utilizado pelo módulo Seletor; (iii) revisar e expandir constantemente o modelo ontológico.

Referências

Allemand, J. N. C. (2006). *Serviço Baseado em Semântica para Descoberta de Recursos em Grade Computacional*. Dissertação (mestrado em ciência da computação), UnB, Brasília, DF.

Chakraborty, D., Perich, F., Avancha, S., and Joshi, A. (2001). Dreggie: Semantic service discovery for m-commerce applications. In *Workshop on Reliable and Secure Applications in Mobile Environment, 20th Symposiom on Reliable Distributed Systems*.

Costa, C. A., Yamin, A. C., and Geyer, C. F. R. (2008). Toward a general software infrastructure for ubiquitous computing. *IEEE Pervasive Computing*, 7(1):64–73.

Dickinson, I. (2009). Jena ontology api. Disponível em: <http://jena.sourceforge.net/ontology/>. Acesso em abril de 2011.

Dilli, R. M. (2010). *Uma Proposta para Descoberta de Recursos na Computação Ubíqua com Suporte Semântico*. Dissertação (mestrado em ciência da computação), UCPEL, Pelotas, RS.

Lopes, J. L. B., Pilla, M. L., and Yamin, A. C. (2007). Exehda: a middleware for complex, heterogeneous and distributed applications. *Iberian-American Conference on Technology Innovation and Strategic Areas*.

McGuinness, D. L. and van Harmelen, F. (2009). Owl web ontology language overview. Disponível em: <http://www.w3.org/TR/owl-features/>. Acesso em abril de 2011.

Prud'hommeaux, E. and Seaborne, A. (2008). Sparql - a query language for rdf. Disponível em: <http://www.w3.org/TR/rdf-sparql-query/>. Acesso em abril de 2011.

Robinson, R. and Indulska, J. (2007). *Resource Discovery in Pervasive Computing Environments*. Handbook on Mobile Ad Hoc and Pervasive Communications. American Scientific Publishers.

Schaeffer, A. E. (2005). *PerDiS: um Serviço para Descoberta de Recursos no ISAM Pervasive Environment*. Dissertação (mestrado em ciência da computação), UFRGS, Porto Alegre, RS.

Soldatos, J., Stamatis, K., Azodolmolky, S., Pandis, I., and Polymenakos, L. (2007). Semantic web technologies for ubiquitous computing resource management in smart spaces. *Int. J. Web Eng. Technol.*, 3(4):353–373.

Tangmunarunkit, H., Decker, S., and Kesselman, C. (2003). Ontology-based resource matching in the grid - the grid meets the semantic web. In Fensel, D., Sycara, K. P., and Mylopoulos, J., editors, *International Semantic Web Conference*, volume 2870 of *Lecture Notes in Computer Science*, pages 706–721. Springer.

Zhu, F., Mutka, M., and Ni, L. (2005). Service discovery in pervasive computing environments. *IEEE Pervasive Computing*, 4(4):81–90.