THIAGO SOUTO MENDES

UM MODELO DE ARQUITETURA DE *MATCHING* ENTRE PERFIS DE USUÁRIOS E SERVIÇOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de "Magister Scientiae".

VIÇOSA

MINAS GERAIS - BRASIL

2008

Ficha catalográfica preparada pela Seção de Catalogação e Classificação da Biblioteca Central da UFV

T
M538m
2008

Mendes, Thiago Souto, 1984-
Um modelo de arquitetura de *Matching* entre perfis de usuários e serviços / Thiago Souto Mendes. – Viçosa, MG, 2008.
xvi, 89f.: il. (algumas col.) ; 29cm.

Orientador: Mauro Nacif Rocha.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f. 83-89.

1. Comunicação móvel. 2. Web semântica. 3. Ontologia. 4. Engenharia de software. I. Universidade Federal de Viçosa. II.Título.

CDD 22.ed. 004.6

THIAGO SOUTO MENDES

UM MODELO DE ARQUITETURA DE *MATCHING* ENTRE PERFIS DE USUÁRIOS E SERVIÇOS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

APROVADA: 28 de novembro de 2008

Alcione Paiva de Oliveira
(Co-orientador)

José Luis Braga
(Co-orientador)

Vladimir Oliveira Di Iorio

Rogério Martins Gomes

Mauro Nacif Rocha
(Orientador)

Aos meus pais, Dalvadísio (*in memoriam*) e Mirian.

Aos meus irmãos, Daniel e Flavia.

Aos meus sobrinhos.

A minha eterna companheira Nidyana Miranda.

À minha Família e Amigos queridos.

AGRADECIMENTOS

A Deus, por sempre estar ao meu lado durante minha vida me dando saúde e força para superar todos os obstáculos e me oferecendo muitos momentos felizes, e sempre colocando pessoas boas ao meu redor.

A minha família, por ser o meu porto seguro e pelo exemplo de união. A minha mãe Mirian e a minha vó Nila, pelo amor, carinho, educação, e por ser minha fonte de inspiração. A meu pai, Dalvadísio (in memoriam) que sempre me deu muito carinho e muito amor e que com certeza está muito feliz com mais essa conquista. A meu irmão, Daniel pelo companheirismo durante todos esses anos e ensinamentos durante minha vida. A minha cunhada, Cintia por estar presente em nossas vidas. A minha irmã e meus sobrinhos, Flávia, Arthur, Luiz Evandro, Marcos Vinicius e João Pedro. Aos meus tios, tias, primos e primas que sempre me deram apoio e torceram por mim.

À Universidade Federal de Viçosa, pela oportunidade em realizar este curso e por ser um lugar maravilhoso por toda sua estrutura e beleza natural.

Ao Professor Mauro Nacif, pela dedicação e orientação e, principalmente, pela preocupação com a formação de um profissional competente.

Ao Professor e Pesquisador Brauliro Gonçalves Leal, pelos grandes ensinamentos, pela amizade, pelo incentivo e pelo exemplo de amor à pesquisa.

Ao Professor José Luis Braga, pela confiança, pelo apoio, pelas sugestões e pelo empenho na realização do trabalho.

Ao Professor Alcione Paiva de Oliveira, pelo apoio, pelo aconselhamento e sugestões.

A todos os professores do DPI/UFV pela amizade e pela valiosa colaboração durante a realização do trabalho e curso.

A Nidyana Miranda, por ser minha eterna e grande companheira. Foi ela quem mais me ajudou a superar todos os momentos difíceis durante o mestrado. E também por estar sempre presente nos momentos felizes no mestrado e no egresso dos meus empregos atuais.

Aos todos os meus amigos, principalmente, Rossini Pena, Sergio, Lissandra, Carlos Alberto, Adelson, Rogério, Flávio Cardeal, Marconi Arruda, Alessandro Oliveira, Edgard Fiuza, Raquel Lana, Ana Carla, Amadeu e Sabrina pelo convívio e demonstrações de carinho e amizade.

Aos colegas de curso e aos muitos amigos das repúblicas onde eu morei durante minha estadia em Viçosa.

Aos funcionários Altino, Paulinho e Eliana, pelo apoio na condução do trabalho.

Aos professores, funcionários e aos meus alunos do DECOM/CEFET-MG, pela amizade e pelo grande incentivo na fase final da realização do trabalho.

Aos meus colegas de trabalho da Belo Horizonte Sistemas, pela força nos meus últimos passos em direção a conclusão desse trabalho.

BIOGRAFIA

Thiago Souto Mendes, filho de Dalvadísio Mendes de Andrade e Mirian Souto Mendes, brasileiro nascido em 13 de julho de 1984 no município de Vitória da Conquista, no Estado da Bahia.

No início de 2002, ingressou no curso de Ciência da Computação na Universidade Vale do Rio Doce em Governador Valadares/MG, onde gradou-se no ano de 2005. Em 2006, mudou-se para a cidade de Viçosa para cursar o mestrado em Ciência da Computação na Universidade Federal de Viçosa – UFV onde se tornou mestre em novembro de 2008. Atualmente reside em Belo Horizonte onde trabalha como professor substituto no Departamento de Computação do CEFET-MG e como analista de sistemas na empresa Belo Horizonte Sistemas.

SUMÁRIO

Página

LISTA DE TABELAS

LISTA DE FIGURAS

LISTA DE SIGLAS

API	Application Programming Interface
ASR	Area Service Register
DAML	DARPA Agent Markup Language
F-Logic	Frame Logic
FOAF	Friend-Of-A-Friend
GPS	Global Positioning System
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IHC	Interface Humano-Computador
IOPE	Inputs Outputs Preconditions Effects
JAVA ME	Java 2 Platform Micro Edition
JAX-RPC	Java API for XML-based Remote Procedure Call
JSP	JavaServer Pages Technology
KB	Knowledge Base
LIF	Location Interoperability Forum
MIDP	Information Device Profile
MIME	Multipurpose Internet Mail Protocol
MVC	Model-View-Controller
NAISC	New Atlantis International Services Corporation
OGC	Open Geospatial Consortium
OpenLS	OpenGIS Location Services
OWL	Web Ontology Language
OWL-S	Web Ontology Language for Services
PC	Personal Computer
PDA	Personal Digital Assistants
PoI	Point of Interest
RCC	Regional Connection Calculus
RMI	Remote Method Invocation
RPCs	Remote Procedure Calls
SBL	Serviços Baseados em Localização
SIG	Sistema de Informação Geográfica

SIG	Sistemas de Informação Geográfica
SLS	Serviço de Localização de Serviço
SOAP	Simple Object Access Protocol
SOCAM	Service-Oriented Context-aware Middleware
SOUPA	Standard Ontology for Ubiquitous and Pervasive Applications
SOMM	Souto Onthology Matching Model
SWRL	Semantic Web Rule Language
UDDI	Universal Description, Discovery and Integration
UNSPSC	United Nations Development Programme
UPnP	Universal Plug and Play
URI	Universal Resource Identifier
URL	Universal Resource Locator
W3C	World Wide Web Consortium
WFS	Web Feature Service
WMS	Web Map Service
WSDL	Web Services Description Language
XML	eXtensible Markup Language

RESUMO

Mendes, Thiago S., M.Sc., Universidade Federal de Viçosa, novembro de 2008. **Um Modelo de Arquitetura de *Matching* de Perfis de Usuários e Serviços Baseados em Localização.** Orientador: Mauro Nacif Rocha. Co-Orientadores: José Luis Braga e Alcione Paiva de Oliveira.

Na era da Computação Pervasiva, uma classe de aplicações vem despertando um interesse crescente na comunidade acadêmica, chamadas aplicações móveis sensíveis ao contexto. A Computação Ciente de Contexto necessita de mecanismos de representação de informações de contexto que facilitem o processamento dessas informações por infra-estruturas de *software*. Uma possível solução é integrar tecnologias de *Web* Semântica no desenvolvimento de aplicações cientes de contexto através da modelagem e implementação de ontologias como extensão a uma infra-estrutura de gerenciamento de informações de contexto. A arquitetura deve possuir um modelo formal e padrão de representação de informações de contexto, suporte à interoperabilidade semântica entre aplicações cientes de contexto e mecanismos de inferência com base na semântica desse modelo ontológico. Este trabalho propõe um modelo de arquitetura distribuída, baseada em dispositivos móveis e orientada a serviços, capaz de anotar semanticamente a descrição dos serviços e do perfil dos usuários. Um usuário poderá fazer buscas e execução de serviços semânticos, além de poder consultar mapas para descobrir a localização de serviços.

ABSTRACT

Mendes, Thiago S., M.Sc., Universidade Federal de Viçosa, november, 2008. **A Model of Architecture of *Matching* of *Profile*s of Users and Location-Based Services.** Adviser: Mauro Nacif Rocha. Co-Advisers: José Luis Braga and Alcione Paiva de Oliveira.

In the era of ubiquitous computing, a new kind of application has been gaining increased attention from the academic community, the so-called *Context-aware* mobile applications. *Context-aware* computing requires context information representation mechanisms that facilitate this type of information processing within appropriate *software* infrastructures. One possible solution is to integrate Semantic *Web* technologies into the development of *Context-aware* applications through ontology modeling and implementation as an extension to the context information's management infrastructure. Such architecture should have a formal model and a context information representation standard, as well as semantic interoperability support between *Context-aware* applications and inference mechanisms based on the ontological model semantics. In this work we propose a distributed architecture model based on mobile devices and service oriented, capable of semantic annotation of service and user profiles. This enables a user to search for and execute semantic services, and also retrieve maps for service location discovery.

1 INTRODUÇÃO

Atualmente, com o aumento do uso dos dispositivos móveis e a ampliação de sua capacidade de processamento, armazenamento e a cobertura oferecida pelas redes de comunicação tornaram-se possível às pessoas utilizarem esses aparelhos durante todo o tempo, proporcionando cada vez mais o desenvolvimento de aplicações mais sofisticadas e capazes de fornecer informações mais personalizadas (Pashtan, 2005).

Esta visão do poder computacional de estar presente em qualquer lugar e hora, realizando serviços de interesse dos usuários, é denominada Computação Ubíqua ou Computação Pervasiva. A Computação Pervasiva tem como principal objetivo fornecer ao usuário um acesso imediato à informação, de forma transparente, e possibilitar a execução de suas tarefas (Weiser, 2008).

Conforme Weiser (2008), futuramente, os computadores estarão distribuídos pelo ambiente físico, contudo eles serão invisíveis ao usuário, deixando-o concentrar-se mais em suas tarefas do que nas ferramentas. Na nova geração da computação, cada pessoa estará interagindo continuamente com diversos computadores próximos, interconectados por uma rede sem fio. Para atingir esse ponto, serão necessários novos tipos de computadores de vários tamanhos e formas.

Uma das formas de se acessar as informações na aquisição de serviços é através da *Web*. Para auxiliar nesse acesso, surgiram os Serviços Baseados em Localização (SBL), que visam fornecer ao usuário informações de acordo com a sua localização, ou seja, informações que pertençam a um domínio do interesse do usuário e que possam ser acessadas quando o usuário retornar à área onde estava inicialmente (Schiller, 2004).

Para os SBL, a posição do usuário é definitivamente o componente preliminar do contexto. Contexto é toda informação que possa influenciar o resultado fornecido aos usuários em resposta às suas perguntas, pois os ambientes móveis são dinâmicos e podem mudar assim que o usuário se movimenta ou faça uma pergunta diferente. Conseqüentemente, o desafio para SBL é monitorar continuamente a relevância semântica dos dados em um contexto (Bharat, 2003).

Segundo Pashtan (2005), o perfil de usuário é a chave de acesso a serviços e informações personalizadas. Caso dois usuários consultem o mesmo serviço, no mesmo local e ao mesmo tempo, o sistema deverá fornecer respostas diferentes de acordo com seus respectivos perfis. O perfil de usuário contém informações sobre os dados pessoais, preferências ou interesses.

Nos SBL, os perfis de usuário são caracterizados por seu dinamismo, contrastando com os perfis de usuário fixos usados principalmente nos serviços de *Web Services*. Dependendo de onde, quando, como, com quem e porque, o perfil do usuário e suas necessidades irão variar. Os perfis dos usuários nos SBL aumentam a relevância semântica dos dados de contexto (Yu et al., 2004).

Uma questão semântica nos SBL é a informação da localização dos dados que os mesmos podem usar para encontrar a resposta para uma consulta do usuário. A localização das fontes de dados disponíveis muda enquanto o usuário se movimenta, sendo caracterizadas por um perfil dos dados.

A fonte de dados dos serviços é identificada pelos SBL usando um mecanismo da descoberta do serviço baseado em *matching* entre a consulta do usuário, que é gerada de acordo com as informações de localização, o perfil do usuário e dos dados. Para auxiliar no *matching*, os SBL podem procurar o conhecimento ontológico relevante (Nakanishi et al., 2004).

1.1 O Problema e sua Importância

Na última década, houve uma tendência de mudança no modelo de computação estático, previsível e baseado em estações de trabalho, para um modelo de computação dinâmico, com mudanças constantes de ambiente ocasionadas pela mobilidade do usuário, e caracterizado pela utilização de dispositivos móveis, como celulares, *Smartphones* e *PDA's,* como mostra a **Figura 1**. Esta mudança deu mais um passo em direção ao conceito de Computação Pervasiva (Weiser, 2008).

Figura 1 - Exemplos de dispositivos computacionais ubíquos.

O desenvolvimento de novas aplicações que explorem mudanças de contexto dentro de um domínio dinâmico é sugerido pela Computação Pervasiva. Ao contrário das aplicações tradicionais, essa nova categoria de aplicações, denominadas “aplicações móveis sensíveis ao contexto”, leva em consideração, em seus processamentos e em sua tomada de decisão, não somente as entradas de dados fornecidas pelos usuários explicitamente, mas também entradas implícitas, relativos ao contexto físico, computacional, dos usuários e os ambientes que os circundam.

As aplicações sensíveis ao contexto e os ambientes ubíquos vêm emergindo como uma alternativa para várias áreas de aplicação, como educação, saúde, turismo, negócios, segurança, entre outras.

Além do mais, trazem novos desafios que precisarão ser enfrentados, como segurança e privacidade, protocolos de comunicação e de redes de sensores e infra-estrutura, utilização eficiente de energia nos dispositivos, mobilidade e heterogeneidade de usuários e dispositivos, interoperabilidade, mudança e adaptação de comportamento de aplicações mediante alterações nos ambientes e melhoria na comunicação e interação entre homens e computadores.

Em um ambiente ubíquo, a manipulação de contexto exige a superação de um conjunto de desafios tecnológicos e de *design*. O desenvolvimento de aplicações móveis sensíveis ao contexto requer uma infra-estrutura de suporte de *software* adequada à manipulação de contextos diversos, provenientes de ambientes altamente dinâmicos, distribuídos, heterogêneos e em constantes mudanças.

Temos presenciado o aparecimento de várias pesquisas relacionadas à construção de plataformas de serviços com suporte à execução dessas aplicações (Nakanishi et al., 2004), (Gu et al., 2004), (Dey e Abowd, 2000). Essas plataformas visam oferecer suporte arquitetural e de programação para que os projetistas desenvolvam suas aplicações utilizando mecanismos, serviços e interfaces que oculte a complexidade da manipulação de contexto.

Quando superadas determinadas limitações das tecnologias sem fio, como por exemplo, largura de banda, roteamento, privacidade, e quando existir um apropriado suporte arquitetural para manipulação de contexto, o universo das potenciais aplicações e dos domínios possíveis de utilização em sistemas sensíveis ao contexto é ilimitado.

Apesar dos estudos, ainda existem questões em aberto, abrindo espaço para desenvolver investigações adicionais, particularmente no que se refere à generalidade e modelagem da informação contextual, à distribuição do processamento através de *Web Services*, o uso de ontologias para armazenar as informações de contexto e o gerenciamento das informações de localização, ou mesmo, ao aproveitamento de tecnologias da *Web* Semântica na descrição dos serviços.

Portanto, este trabalho tem o propósito de ampliar a análise do contexto do usuário através do uso de ontologias, considerando também o contexto do ambiente para saber em que situação o cliente se encontra gerenciando as informações de localização. Outro propósito será o suporte à expansão da arquitetura, pois ela poderá ser implementada por linguagens de programação diferentes. Por isso, os serviços serão distribuídos através do uso de *Web Services* permitindo que a carga de processamento seja divida entre os serviços, além de possibilitar que os usuários acessem apenas os serviços necessários.

1.2 Objetivos

O objetivo deste trabalho é a criação de um modelo conceitual de arquitetura de serviços que permita a identificação da semelhança dos interesses dos

usuários e dos serviços existentes que estejam em uma mesma localização geográfica. Detalhadamente, os objetivos específicos são:

- Propor um modelo de arquitetura genérica que possa ser usado como base para o projeto de *software*s que sejam orientados a contexto para ambientes móveis;
- Criar modelos de perfis de usuários e de serviços, bem como descrever este modelo em uma linguagem de representação que possibilite algum tipo de inferência sobre os mesmos;
- Desenvolver um serviço de *matching* para disponibilizar informações personalizadas para o usuário de acordo com seus interesses e sua localização;
- Desenvolver um protótipo de sistema baseado na arquitetura proposta, para o Departamento de Informática da Universidade Federal de Viçosa (DPI/UFV) como forma de validação e teste da arquitetura.

1.3 Organização deste Documento

Esta dissertação está dividida da seguinte forma:

No Capítulo II é apresentada toda a fundamentação teórica sobre os sistemas cientes de contexto, apresentando as definições sobre as informações sobre contexto e Serviços baseados em localização.

O Capitulo III apresenta as tecnologias de *Web Services* Semânticos e tecnologias relacionadas, fazendo uma pequena descrição sobre os tipos de ontologias e busca semântica.

O Capítulo IV tem por objetivo descrever os principais trabalhos relacionados à construção de sistemas para Computação Pervasiva.

O Capitulo V, é a parte principal da dissertação, nele são descritas as funcionalidades e requisitos da arquitetura proposta através de diagramas.

Finalmente, no Capítulo VI conclui este trabalho apresentado os resultados alcançados, as contribuições, limitações e dificuldades, bem como, as possibilidades de trabalhos futuros na área.

No final do documento, ainda são apresentados os Apêndices e as Referências Bibliográficas.

2 SISTEMAS CIENTES DE CONTEXTO E SERVIÇOS BASEADOS EM LOCALIZAÇÃO

A modernização dos dispositivos móveis permite que os usuários possam se deslocar enquanto executam as suas tarefas, ou seja, enquanto o usuário consulta informações através de dispositivos móveis a qualquer hora em qualquer lugar. No entanto, além da localização do usuário, outras informações podem ser percebidas e processadas por um serviço remoto graças ao avanço na tecnologia de sensores.

Através do avanço dessas tecnologias, estão surgindo cada vez mais aplicações capazes de produzir informação personalizada de forma transparente ao usuário. Além da localização, é possível realizar buscas semânticas de serviços para lhe oferecer as informações sobre o serviço que ele está sendo requisitado. Nas seções a seguir, são apresentadas as características desse novo paradigma da computação.

2.1 Contexto

De acordo com Arbter (2004), entender o que é contexto e como ele pode ser usado permitirá a construção de aplicações mais eficientes e a melhor escolha do contexto a ser usado no desenvolvimento de aplicações.

Várias publicações apresentam diferentes definições para contexto (Schilit et al., 1994), (Brown et al., 1997). Conforme Dey e Abowd (2000), contexto é qualquer informação que pode ser usada para caracterizar uma situação de uma entidade. Uma entidade é uma pessoa, um lugar, ou um objeto que é considerado relevante para a interação entre um usuário e uma aplicação, incluindo o próprio usuário e a própria aplicação. Chen e Korts (2000) apresentam uma definição que engloba todas as outras definições:

> "*Contexto é o conjunto de estados do ambiente e configurações que determinam um comportamento da aplicação, incluindo o próprio evento da aplicação que ocorre e é de interesse do usuário*"

A mobilidade do usuário cria situações nas quais o contexto é mais dinâmico, ou seja, as informações do contexto mudam constantemente, tornando complicado para o usuário passar todas as informações do contexto para o computador a cada situação. Dessa forma, o computador deve coletar informações do contexto automaticamente de forma transparente para o usuário (Dey e Abowd, 2000).

A classificação das informações de contexto pode ajudar os desenvolvedores a descobrir que tipos de informações de contexto serão úteis para sua aplicação. Segundo Feng, Apers e Jonker (2004), as informações de contexto podem ser classificadas em duas categorias:

1) **Contexto do usuário**: é toda informação relacionada ao usuário. Esse contexto é ainda dividido em:

a) **Perfil:** representa o perfil do usuário, tais como livros, prediletos, músicas preferidas, comida predileta, e outras informações subjetivas do usuário;

b) **Comportamento dinâmico**: representa as tarefas a serem cumpridas durante o dia, semelhante a uma agenda;

c) **Estado Fisiológico**: estas informações são relevantes para o monitoramento da saúde e são obtidos através de sensores que são colocados no corpo do usuário.

d) **Estado Emocional**: esse contexto pode ser capturado através de câmeras (análise visual), interpretação de sinais acústicos, batimento cardíaco ou fornecido manualmente pelo usuário.

2) **Contexto do ambiente**: são informações relativas ao ambiente. Esse contexto é dividido em:

a) **Ambiente Físico**: representa características do ambiente físico, tais como, localização, temperatura, tempo, nível de luminosidade e nível de ruído;

b) **Ambiente Computacional**: representa o ambiente computacional, tais como, largura de banda da rede, capacidade da rede, dispositivos ao redor, impressoras e computadores.

c) **Ambiente Social**: representa o ambiente social, como congestionamento de trânsito, a localização de pessoas ao redor, informações de descontos de lojas, entre outros;

O contexto nos SBL agrupa toda a informação que caracteriza a situação de uma entidade, bem como o significado dos dispositivos, e que pode ser usada para fornecer serviços mais relevantes ao usuário. Os SBL de informações contextuais incluem tipicamente a posição do usuário, o tempo, as condições do tráfego, etc. Os *software*s que utilizam informações de contexto são chamados sistemas cientes de contexto (Held, 2002).

Para Pashtan (2005), tudo é dependente de contexto, particularmente o significado dos termos que aparecem em pedidos de usuário ou em descrições do serviço e dos dados. Contudo, a tecnologia tradicional de banco de dados ignora a maior parte das informações de contexto, fornecendo os dados dos serviços baseados em uma única informação (contexto implícito).

Em um ambiente móvel, a informação contextual pode mudar rapidamente. Para os SBL, as condições da rede, por exemplo, podem mudar e envolver uma estratégia diferente na busca da informação, ou o usuário pode mudar a atividade, do trabalho ao lazer, por exemplo, envolvendo exigências de informação diferentes.

Conseqüentemente, os contextos não podem ser recuperados uma única vez, mas têm que ser monitorados a fim de manter a informação contextual atualizada (Spiekermann, 2004).

O desafio semântico na gerência do contexto dos SBL é determinar quais informações contextuais são relevantes para a tarefa atual. Não faz sentido controlar contextos muito sofisticados se somente alguns componentes forem realmente úteis. Inversamente, serviços de qualidade mais baixa são oferecidos se faltar informação contextual apropriada, determinando o que é relevante tanto no lado do usuário como no lado dos serviços disponíveis para que o *matching* seja feito da melhor forma possível (Pashtan, 2005).

2.1.1 Perfil do Usuário

Os perfis dos usuários são de grande importância para fornecer SBL inteligentes e personalizados. Segundo Chen (2004), através da aquisição do perfil do usuário é possível identificar as preferências do usuário e utilizá-las para melhorar a filtragem das informações e tomadas de decisão.

A maioria das propostas de *Web Services* é personalizada para algum ambiente computacional específico ou aplicação pré-definida. Em serviços baseados em localização, devido à estrutura inerente da mobilidade, o ambiente computacional está mudando continuamente, assim como o tipo e a função de fonte de dados disponíveis. Além disso, o próprio perfil do usuário evolui freqüentemente por causa de fatores diversos, tais como a mudança na posição do usuário, do ambiente social, e da atividade do usuário. Sendo assim, os SBL têm que focalizar em técnicas mais genéricas e mais dinâmicas, sendo capazes de ajustar seu serviço ao perfil atual do ambiente e de usuário (Sinner, 2004).

O perfil do usuário é composto pela informação personalizada das preferências, e em atividades do usuário. A compreensão e as definições das propriedades do perfil do usuário dependem da cultura, da língua, da instrução, etc. Para o usuário em movimento, é incômodo adaptar seu perfil de acordo com línguas ou hábitos locais.

Conforme Bahrat (2003), a utilização de ontologias em SBL pode ser uma boa solução desde que forneçam definições comuns gerais e compartilhadas. A palavra Ontologia é um termo vindo da filosofia que se refere à ciência de descrever as entidades no mundo e como elas estão relacionadas.

Em Ciência da Computação, as ontologias foram primeiramente desenvolvidas na Inteligência Artificial para facilitar o compartilhamento e o reuso do conhecimento, e têm sido estudadas por várias áreas como: Engenharia do Conhecimento, Processamento de Linguagem Natural e Representação do Conhecimento (Davies et al., 2003). Com isso, as ontologias são consultadas para ajudar usuários criar suas próprias hierarquias do conceito (incluindo o perfil do usuário) para o *Web* Site que o usuário esteja navegando (Broens, 2004).

Além do perfil do usuário, deve-se armazenar e tratar o histórico do usuário, isto é, o traço dos serviços já usados pelo usuário, que pode ser de grande utilidade. Conseqüentemente, os SBL devem armazenar o histórico ou “o conhecimento acumulado” do usuário (Nakanishi et al., 2004).

2.1.2 Perfil dos Dados

Os perfis dos dados descrevem os serviços disponíveis. Da mesma maneira que um esquema descreve uma base de dados, um perfil de dados dá a informação sobre os dados fornecidos por um serviço (Held et al., 2002).

Os SBL devem ter suporte a serviços desconhecidos a priori, e cada um deles pode ter sua própria descrição. É uma questão crítica para os SBL entender quais serviços estão disponíveis e onde estão localizados. As técnicas tradicionais da integração de serviços não resolvem o problema porque não cumprem a necessidade de uma resposta rápida e flexível. O uso de ontologias melhoraria a compreensão comum dos serviços, e também permitiria o raciocínio semântico (Weibenberg et al., 2004).

2.1.3 Matching Semântico.

A maioria dos mecanismos existentes para descoberta dos serviços é baseada em palavras-chave, isto é, recuperam as descrições dos serviços que contêm palavras-chave extraídas da pergunta do usuário. Dessa forma, os resultados obtidos costumam ter baixa recuperação (faltando alguns itens relevantes) e baixa precisão (alguns itens recuperados são irrelevantes) (Broens, 2004).

A razão para o primeiro problema é que as palavras-chave podem ser semanticamente similares, mas sintaticamente diferentes, por exemplo, "língua" e "idioma" (sinônimos). A razão para o segundo problema é que as palavras-chave podem ser sintaticamente as mesmas, mas semanticamente podem ser diferentes, por exemplo, "*manga*" (de camisa) e “*manga*” (fruta) (diferentes). Uma solução possível é usar a recuperação baseada em ontologias. Em serviços baseados em localização, o *matching* deve envolver não somente os dados dos serviços, mas também o contexto e o perfil do usuário (e possivelmente o histórico do usuário), a fim de fornecer ao usuário uma informação relevante (Paolucci, 2004).

Dada uma pergunta e dados atualizados dos serviços disponíveis, uma primeira etapa seria filtrar aqueles serviços que não combinam com o tipo do serviço, como por exemplo, se o usuário procurar notícias do curso de Ciência da Computação, as notícias do curso de Letras não devem ser consideradas. Se for feita apenas a combinação das palavras-chave, possivelmente poderão faltar dados de alguns serviços. Por exemplo, podem faltar as notícias do curso de Sistemas de Informação. Para isso se faz necessário o uso de terminologias e ontologias para ajudar a fazer o *matching* das informações.

A segunda etapa reescreveria a pergunta fazendo-a mais precisa, usando o perfil do usuário (Sinner, 2004). A questão é que propriedades do perfil do usuário devem ser usadas nesta etapa. Uma solução possível é fazer um exame de propriedades no perfil do usuário que “correspondem” às propriedades no perfil dos dados.

A terceira etapa reescreveria a pergunta, adicionando circunstâncias em relação ao contexto do ambiente no qual o cliente se encontra. Outra vez, não é fácil decidir quais contextos devem ser considerados. Uma solução possível é olhar atributos espaço-temporais no perfil dos dados. A pergunta é combinada finalmente com o perfil dos dados e os dados dos serviços que restaram (Yu et al., 2004).

2.2 Computação Ciente de Contexto

Para Dey e Abowd (2000), um sistema ciente de contexto, do inglês *Context-aware*, é aquele que usa o contexto para prover informações relevantes ou serviços ao usuário.

São consideradas aplicações cientes de contexto aquelas que têm seu comportamento modificado de acordo com as informações do contexto ou aplicações que simplesmente mostram ao usuário informações do contexto.

As aplicações cientes de contexto podem ser dividas em três categorias (Dey e Abowd, 2000):

- **Adaptação ao contexto** (*contextual adaptation*): é a habilidade de executar ou modificar um serviço automaticamente, baseando-se

no contexto atual. As aplicações baseados numa simples regra de *if-then-else*. Um comando é executado quando existe certa combinação de contexto;

- **Sensoriamento do contexto** (*contextual sensing*): é a habilidade de detectar informações sobre o contexto e apresentá-las ao usuário. Nesta categoria, estão as aplicações que buscam informações para o usuário sobre o contexto;
- **Aumento do contexto** (*contextual augmentation*): é a habilidade de associar informações ao contexto para recuperar posteriormente. São aplicações que permitem associar dados digitais ao contexto do usuário.

2.3 Serviços Baseados em Localização - SBL

Os serviços baseados em localização são definidos em Spiekermann (2004), como os serviços que integram a posição de um dispositivo móvel ou a unem com outra informação para fornecer o valor adicional para um usuário. Apesar de essa definição ser um pouco limitada, ela inclui os serviços tradicionais da base de dados que dariam respostas dependendo da posição aos pedidos do usuário.

É preferível pensar em SBL como os serviços que usam principalmente o conhecimento descentralizado, aquele que está geograficamente perto da posição do usuário e conseqüentemente disponibilizar a informação que o usuário em movimento está querendo adquirir, para decidir sobre seu comportamento no curto prazo (dentro das horas ou dos dias seguintes).

A localização é uma característica essencial dos SBL que os distingue da recuperação de informação clássica dos *Web Services*. Assim, um componente necessário na arquitetura dos SBL é uma máquina que guarde a posição do usuário, suportando o armazenamento de dados espaço-temporais.

Os SBL herdam também um número de questões relativas à Computação Pervasiva, tais como a manipulação das autorizações (verificar os parâmetros do dispositivo, empacotar a conexão da rede sem fio, validar

a identificação de usuário), proteção à privacidade do usuário (agora incluindo dados sobre posições e trajetórias do usuário), técnicas de Interface Humano-Computador (IHC) para interagir com os usuários que usam dispositivos com potencialidades limitadas da exposição, entre outras (Spiekermann, 2004).

Do ponto de vista semântico, a característica e o desafio principal de um SBL é o fato de que ele serve como *middleware* entre um usuário possivelmente desconhecido e as fontes dos dados desconhecida a priori. Além disso, a mediação não pode ser preparada adiantadamente, porque os sócios no mediador não são conhecidos antecipadamente (Gu et al., 2004).

Poderíamos dizer que os SBL têm que executar a avaliação da pergunta em um ambiente hostil. Para superar esta dificuldade, as contribuições da maioria das técnicas avançadas são bem-vindas. Incluem: auxílio da ontologia para compreender o que realmente está próximo, busca da informação de ponto a ponto aumentando as possibilidades de encontrar a informação relevante, manipulação da informação incompleta lidando com os dados faltantes, e técnicas de aproximação, determinando o que poderia ser uma resposta razoável quando a combinação perfeita não for possível.

O *Location Interoperability Forum* (LIF) (Anderson e Cheng, 2003) tem dividido SBL em três categorias:

- ***Basic Service level***: implica no uso de tecnologia baseada em células. A acurácia é pobre, mas existem vários terminais computacionais disponíveis que suportam esse nível de serviço. As aplicações que podem usufruir desse tipo de serviço são aquelas nas quais a precisão da localização não é tão importante, como por exemplo, localização de pontos turísticos e entretenimento;
- ***Enhanced Service level***: são aplicações que requerem maior acurácia (cerca de dez metros), como por exemplo, um serviço que rastreia a localização de uma pessoa, anúncios de produtos, entre outros;
- ***Extended Service level***: são aplicações que precisam de alta acurácia (cerca de poucos metros), por exemplo, aplicações de navegação passo-a-passo, nas quais um usuário pode chegar ao

ponto destino através das instruções que são mostradas no seu dispositivo.

Conforme Schiller (2004), o projeto de aplicações SBL pode ser classificado em dois tipos de serviços:

- **Serviço *push*** – implica que o usuário recebe informação sem explicitamente ter requisitado. A informação pode ser enviada ao usuário com ou sem o seu consentimento. Em outras palavras, em uma aplicação *push*, a informação é entregue ao consumidor sem que este tenha controle de quando isto ocorre;
- **Serviço *pull*** – o usuário busca informação sempre que é preciso. Esta informação pode envolver localização, como por exemplo, consultar qual o cinema mais próximo.

É possível observar que a transparência e comodidade oferecidas pelos serviços *push* têm um alto preço. Aplicações *push* são muito caras comparadas aos serviços *pull*. Por exemplo, os serviços *pull* requerem maior quantidade de recurso de rede porque a localização do usuário precisa ser atualizada constantemente. Outra desvantagem dos serviços *push* é a questão da privacidade. A própria noção de atualização constantemente da posição dá a idéia de rastreamento em tempo real (Schiller, 2004).

2.4 Considerações Finais

Neste capítulo, foi abordada uma parte da fundamentação teórica relacionada ao tema desta dissertação. Primeiramente, foi apresentada uma definição mais geral do que é contexto e as formas de categorizá-la segundo Feng, Apers, Jonker (2004) e Dey e Abowd (2000). As informações do contexto são obtidas principalmente através de sensores, destacando o uso de sensores de localização, e capturam as informações do usuário, requisitos essenciais em aplicações SBL.

No próximo capítulo posterior, será abordada a adoção de *Web Services* Semânticos na Computação Pervasiva.

3 UTILIZAÇÃO DE *WEB SERVICES* SEMÂNTICOS NA COMPUTAÇÃO PERVASIVA

A arquitetura de *Web Services* definida pelo *World Wide Web Consortium* (*W3C*, 2008) é baseada na linguagem *XML,* de modo a permitir interoperabilidade entre diferentes plataformas. As mensagens escritas em *XML* são transmitidas utilizando o protocolo *HTTP,* o que facilita o tráfego de mensagens pela Internet.

Essa arquitetura permite a construção de sistemas distribuídos com baixo acoplamento, utilizando tecnologias baseadas em padrões abertos, independentes da linguagem de programação utilizada e da plataforma computacional (Endrei et al, 2004).

A arquitetura de *Web Services* é composta por várias tecnologias dispostas em camadas, como ilustra a **Figura 2**. Dessas tecnologias, se destacam o *Simple Object Access Protocol* (*SOAP*) na camada de mensagem e o *Web Services Description Language* (*WSDL*) na camada de descrição. E na camada de processos, o *Universal Description, Discovery and Integration* (*UDDI*).

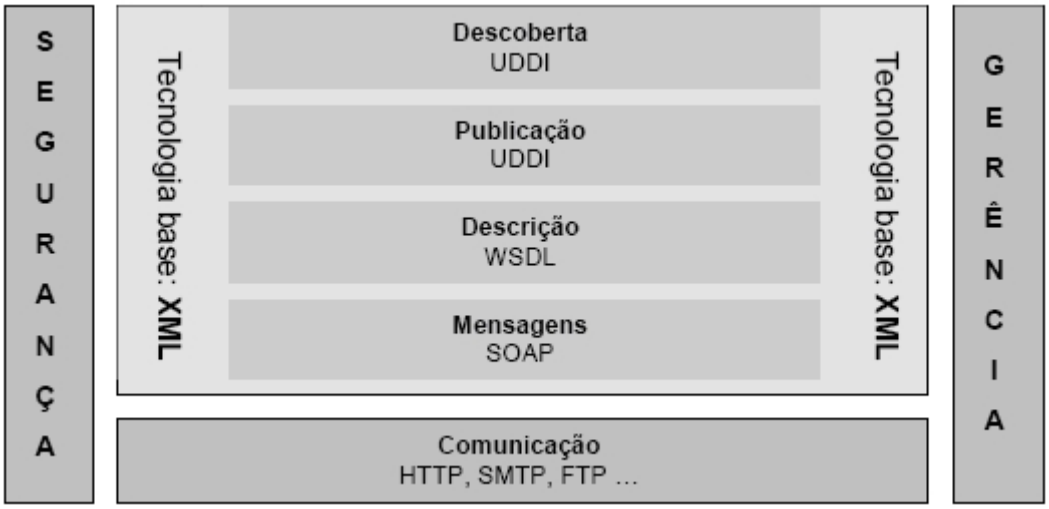


Figura 2 - Tecnologias no desenvolvimento de um Web Service (Souza, 2004).

Este capítulo divide-se da seguinte forma: é apresentada a tecnologia de *Web Services*, em seguida, são apresentadas as principais abordagens de linguagens para descrição e anotação semântica de serviços *Web*. Por

último, são apresentados os principais trabalhos relacionados com a descoberta e invocação automática de serviços semânticos que é um dos focos dessa dissertação.

3.1 Web Sevices

O *Web Service* é definido como um componente de *software*, ou uma unidade lógica de aplicação, que se comunica através de tecnologias e padrões de Internet (Basiura et al, 2003). Esse componente provê dados e serviços para outras aplicações. Esta tecnologia combina os melhores aspectos do desenvolvimento baseado em componentes e a *Web*. Como componentes, representam uma funcionalidade implementada em uma "caixa-preta", que pode ser reutilizada sem a preocupação de como o serviço foi implementado. As aplicações acessam os *Web Services* através de protocolos e formatos de dados padrões, como *HTTP*, *XML* e *SOAP* (Dextra, 2008).

Os *Web Services* conectam aplicações diretamente entre si, e têm como idéia básica que essa conexão se dá sem que seja necessário efetuar grandes alterações nas próprias aplicações. Além disso, uma das premissas fundamentais é que o padrão usado pelas conexões seja aberto e independente de plataforma tecnológica ou linguagens de programação.

3.2 Arquitetura Orientada a Serviço (SOA)

O *SOA* é um tipo de arquitetura de sistemas distribuídos nas quais aplicações utilizam serviços disponíveis em uma rede como, por exemplo, a *Web*. Um serviço fornece uma função específica, tipicamente uma função de negócios como, por exemplo, o processamento de uma ordem de compra. Um serviço pode fornecer uma função simples, como a conversão de um valor de uma moeda para outra, ou ele pode realizar um conjunto de funções de negócios, como o tratamento de diversas operações em um sistema de reserva de passagens aéreas (Oellermann, 2001).

Serviços que realizam um conjunto de funções de negócios, em oposição a uma simples função, são considerados de alta granularidade. Um conjunto múltiplo de serviços pode ser utilizado para satisfazer um requisito

de negócios mais complexo. Na realidade, uma forma de se visualizar *SOA* é a de uma abordagem para a conexão de aplicações (expostas como serviços), de tal forma que elas possam comunicar umas com as outras. Resumindo, *SOA* é uma forma de compartilhar funções (tipicamente, funções de negócios) de uma forma ampla e flexível.

A arquitetura de comunicação de um *Web Service* está definida nas interações de três papéis: provedor, solicitante e um servidor de registro, ilustrados na **Figura 3**.

Figura 3 - Arquitetura de *Web* Services (Ferris e Farrell, 2003).

O provedor de serviços é o responsável por disponibilizar os serviços e armazenar sua descrição, através da *WSDL*, a qual contém detalhes de *interface*, operações dos serviços e mensagens de entrada e saída para as operações (Souza, 2004). Sendo assim, após o recebimento da descrição de um serviço, o provedor publica a descrição do mesmo, em *WSDL*, em um registro de serviços, ou seja, o solicitante do serviço é uma aplicação que invoca uma interação com o serviço, a qual pode ser um navegador *Web* ou outra aplicação qualquer como outro *Web Service*. Já o servidor de registro é o local onde os provedores publicam os seus serviços, os quais são procurados pelos solicitantes.

3.2.1 Tecnologias

As tecnologias utilizadas em *Web Services* permitem que os serviços possam ser disponibilizados na *Web* de forma padronizada (Souza, 2004), as quais, baseadas em *XML*, são utilizadas para transportar e transformar

dados entre aplicações. Desta forma, as tecnologias utilizadas no desenvolvimento de um *Web Service* podem ser representadas conforme a **Erro! Fonte de referência não encontrada.**.

Assim, a linguagem *XML* é a base para desenvolvimento de um *Web Service*, pois provê meios para a definição e processamento de dados. Os serviços são invocados e fornecem resultados através da troca de mensagens, empacotadas através do protocolo *SOAP*, o qual provê um formato de serialização, utilizado para transmitir documentos em uma rede de comunicação e uma convenção para representar interações entre *Remote Procedure Calls* (RPCs) (Newcomer, 2002). Sendo assim, o uso de *XML* é fundamental para que se garanta a interoperabilidade.

Para promover interoperabilidade entre sistemas heterogêneos é necessário um mecanismo que permita que a estrutura e o tipo de dados possam ser compreendidos pelos *Web Services*. *WSDL* é utilizada com este objetivo, pois permite que mensagens com a descrição precisa dos serviços possam ser trocadas (Booth, 2003). Já o registro *Universal Distribution Discovery and Integration* (*UDDI*) é utilizado para publicação e descoberta de informações sobre *Web Services* (Newcomer, 2002).

3.2.1.1 Extensible Markup Language (XML)

No contexto dos *Web Services*, a *XML* não é apenas utilizada como um formato para a troca de mensagens, mas também como a forma através da qual os serviços são definidos. Como conseqüência, é importante conhecê-la na forma como ela é utilizada para definir e implementar os *Web Services* (Newcomer, 2002). Usando a *XML*, pode-se definir qualquer número de elementos (*tags*), representados entre os sinais de “<” e “/>”, que associam significado às informações. Considere a expressão ilustrada na **Figura 4**.

```
<?xml version="1.0"?>
<aviso>
<para>Mirian data="16/06/2008"</para>
<de>Thiago</de>
<cabecalho>Lembre-se</cabecalho>
<corpo>Te amo Mãe!</corpo>
</aviso>
```

Figura 4 - Exemplo de arquivo *XML*.

Neste exemplo, a *XML* não apenas demonstra elementos que descrevem dados, como também uma estrutura que relaciona um grupo de dados, o que torna fácil imaginar a busca por um elemento que satisfaça a certos critérios, tal como destinatário (<para>) para um determinado indivíduo. Além disso, esquemas associados a um documento em *XML* validam os dados separadamente e descrevem outros atributos relacionados a estes dados.

Desta forma, duas partes que troquem dados em *XML* poderão entender e interpretar os elementos da mesma forma somente se elas compartilharem a mesma definição, ou seja, se as duas partes que compartilharem o mesmo *XML* também compartilharem o mesmo esquema, elas poderão entender o significado dos elementos entre as *tags* igualmente, que é a forma exata como os *Web Services* trabalham (Newcomer, 2002).

A sintaxe de *XML* usada em *Web Services* especifica como os dados são representados, define como os dados são transmitidos, e detalhes de como os serviços são publicados e descobertos. Considere o arquivo em *XML* da **Figura 5**, em que para validação do documento abaixo, torna-se necessário a utilização de um esquema *XML*.

```
<?xml version="1.0" encoding="iso-88591"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:element name="recado">
        <xs:complexType>
                <xs:sequence>
                        <xs:element name="para" type="xs:string"/>
                        <xs:element name="de" type="xs:string"/>
                        <xs:element name="cabecalho" type="xs:string"/>
                        <xs:element name="corpo" type="xs:string"/>
                </xs:sequence>
        </xs:complexType>
</xs:element>
```

Figura 5 - Schema *XML*.

O exemplo ilustra um esquema que define a validação do arquivo em *XML*, os tipos de dados correspondentes a cada elemento, e a estrutura do documento para cada registro de usuário.

A primeira declaração do esquema (primeira linha) referencia o esquema em *XML* que está em uso através do <namespace www.w3.org/2004/xml*Schema*>, mecanismo este que é utilizado

constantemente pelo *W3C* (*W3C*, 2008) para identificar versões aplicáveis. O esquema também contém os tipos de dados (corpo). O *parser XML* valida instâncias de documentos de acordo com os nomes de elementos declarados no respectivo esquema, e elementos que não foram declarados podem ser rejeitados.

3.2.1.2 XML Schemas

Um *schema* (esquema) é uma especificação formal da gramática utilizada por um documento *XML* específico. Ou seja, é utilizado para validar os documentos *XML*, quando estes estão de acordo com a gramática expressa pelo *schema*, permitindo assim, a troca de dados entre aplicações (Coyle, 2002).

3.2.1.3 Web Service Description Language (WSDL)

A *WSDL* é uma linguagem padrão *XML* que foi criada para descrever e publicar os formatos e protocolos de um *Web Service*, e cujos elementos contêm a descrição dos dados, das operações que podem ser realizadas com esses dados e informações sobre o protocolo de transporte que será utilizado (Sommerville, 2007).

Projetada para ser analisada como qualquer outro documento *XML*, a *WSDL* é altamente flexível e extensível. Assim, se o remetente e o destinatário da mensagem puderem compartilhar e entender arquivos *WSDL* da mesma forma, então a interoperabilidade pode ser assegurada (Souza, 2004).

A *WSDL* é dividida em três elementos principais: definições de tipo de dados, operações abstratas e protocolos de ligação. Cada um desses elementos pode ser especificados em documentos *XML* diferentes e importados em diferentes combinações para criar a descrição final de um *Web Service*, ou definidos juntos em um único arquivo *XML*. A definição de tipo de dados determina a estrutura e o conteúdo das mensagens, sendo que as operações abstratas determinam as operações possíveis, e o protocolo de ligação determina a forma de transmissão das mensagens pela rede até os destinatários.

O uso da *WSDL* permite a divisão da descrição dos serviços básicos em duas partes (*interface* e implementação do serviço) como mostrado na

Figura 6, o que permite que estas partes possam ser utilizadas separadamente (Souza, 2004).

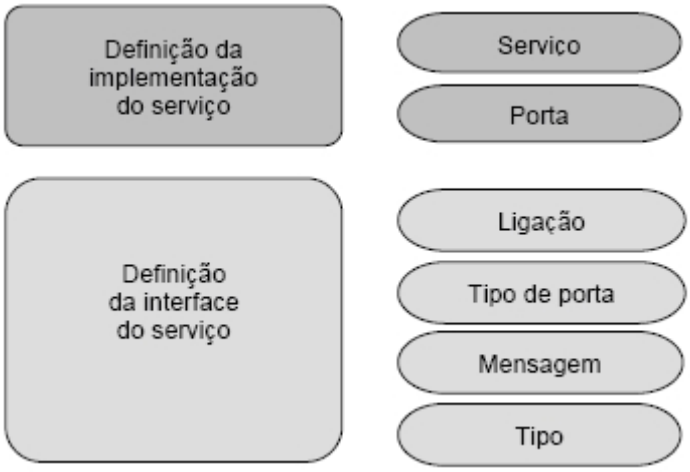


Figura 6 - Camada de descrição de serviços (Souza, 2004).

A implementação do serviço, definida através de um documento *WSDL*, descreve como uma *interface* é implementada por um provedor. Para isso, um arquivo de implementação descreve onde o *Web Service* está instalado e como este é acessado. Além das definições de *interface* e implementação, a *WSDL* especifica extensões para ligação com protocolos e formatos de mensagem, como *SOAP*, *HTTP* e *MIME* (Hansen, 2003).

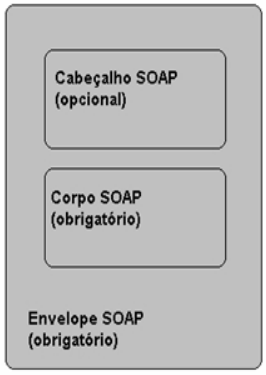
Uma *interface* do serviço contém a definição *WSDL*, o que possibilita que esta seja utilizada, instanciada e referenciada por múltiplas definições de implementação de serviços. A ligação descreve o protocolo, o formato dos dados, a segurança, e outros atributos para a *interface* de um serviço particular; no tipo de porta, os elementos das operações do *Web Service* são definidos; a mensagem é utilizada para definir os parâmetros de entrada e saída de dados de uma operação; e o tipo define o uso de tipos de dados complexos dentro de uma mensagem (Hansen, 2003).

3.2.1.4 Simple Object Access Protocol (SOAP)

O *SOAP* é um protocolo utilizado na troca de informações em um ambiente descentralizado e distribuído, permitindo que isso seja feito entre diversas aplicações independente de sistema operacional, linguagem de programação ou plataforma (Newcomer, 2002). Isso porque a comunicação é feita através de trocas de mensagens, transmitidas em formato *XML*, incluindo parâmetros usados na chamada, bem como campos de resultados,

o que significa que as mensagens podem ser entendidas por quase todas as plataformas de *hardware*, sistemas operacionais, linguagens de programação ou *hardware* de rede. Pode ser utilizado também para invocar, publicar e localizar *Web Services* no registro *UDDI* (Hansen, 2003).

O pacote *SOAP* constitui-se de três partes, que podem ser verificadas na **Figura 7** (Seely, 2002):




```
<env:Envelope xmlns:env="">
<env:Body>
<m:getLastTradePrice env:encodingStyle =
"http://www.w3.org/2001/06/soap-enconding"
xmlns:m="http://www.w3.org/2001/06/quotes"
<symbol>DIS</symbol>
</m:getLastTradePrice>
</env:Body>
</env:Envelope>
```

Figura 7 - Partes de uma mensagem *SOAP*.

• **envelope *SOAP***: define o início e fim da mensagem, quem pode processá-la e se o tratamento é obrigatório ou opcional;

• **codificação *SOAP***: define os mecanismos de serialização que podem ser usados para a troca de instâncias ou tipos de dados por uma aplicação;

• **RPC *SOAP***: especifica como o modelo RPC interage com o *SOAP*, com o objetivo de invocar procedimentos em um sistema remoto.

3.2.1.5 Universal Description, Discovery and Integration (UDDI)

O *UDDI* consiste em uma especificação técnica para descrever, descobrir e integrar *Web Services* como mostra a **Tabela 1**, e é constituído de duas partes: uma especificação técnica para construir e distribuir *Web Services*, através das quais as informações são armazenadas em um formato *XML* específico; e, o *UDDI Business Registry,* que é uma implementação operacional completa da especificação *UDDI* (Sommerville, 2007).

Tabela 1- Operações *UDDI* e suas descrições (Souza, 2004).

Informação	**Operações**	**Informações Detalhadas** (suportadas pela *API* de baixo nível)
White Pages: Informações sobre nome, endereço, número de telefone, e qualquer outra informação de contato de um determinado negócio,	***Publish***: Como o provedor de um serviço *Web* se registra.	***Business information***: Contida em um objeto *BusinessEntity* , que a partir do seu relacionamento com outros objetos pode-se obter informações sobre serviços, categorias, contatos, *URLs*, e outros dados necessários para se interagir com um determinado negócio.
Yellow Pages: Informação que categoriza negócios. Esta categorização está baseada em padrões existentes (não-eletrônicos).	***Find***: Como encontrar um serviço *Web* em particular.	***Service information***: Descreve um grupo de serviços *Web*. Esses estão contidos em um objeto *BusinessService.*
Green pages: informações técnicas sobre os serviços *Web* de um determinado negócio.	***Bind***: Como a aplicação se conecta e interage com um serviço *Web*, após ele ter sido localizado. ***BindingTemplate***	***Binding information*:** Se obtêm os detalhes técnicos necessários para se invocar um serviço *Web*. Isso inclui *URLs*, informações sobre os nomes dos métodos, tipos de argumento, e assim por diante. O objeto *BindingTemplate* representa esses dados. ***Service Specification Detail***: Descreve meta dados sobre as várias especificações implementadas por um determinado serviço *Web*. Na especificação *UDDI* são chamados de *tModels.*

As entidades *UDDI* provêem suporte para descrever a informação sobre negócios e serviços, e como a informação presente no *WSDL* complementa a informação presente no *UDDI*. O processo de registro das informações ocorre como demonstrado na **Figura 8** (Newcomer, 2002).

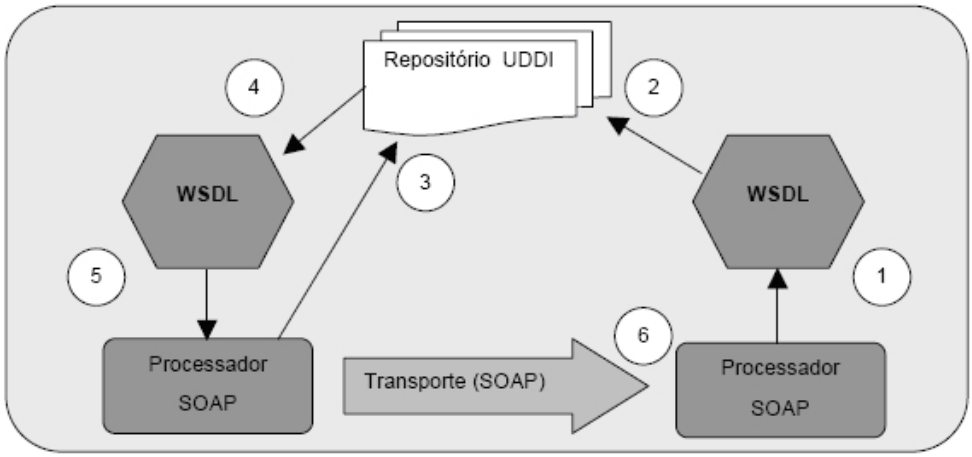


Figura 8 - *UDDI* utilizado para descobrir um *Web Service* (Newcomer, 2002).

Sendo assim, gera-se primeiramente o arquivo *WSDL* para descrever o *Web Service* com suporte do processador *SOAP* (1) e utiliza-se a API *UDDI* para registrar as informações no repositório (2). Depois disso, os dados são transmitidos junto com informações sobre contato, e o registro possui uma entrada com uma URL que aponta para o Servidor *SOAP* com a localização do *WSDL* ou outro documento com a descrição do *Web Service*. Então, outro processador *SOAP* requisita o registro (3) para obter o *WSDL* (4) e, finalmente o cliente gera a mensagem apropriada (5) para enviar a uma operação específica através de determinado protocolo (6). Cabe salientar que o cliente e o servidor devem estabelecer o mesmo protocolo (nesse exemplo, *SOAP* sobre *HTTP*) e compartilhar a mesma semântica para a definição do serviço, a qual, neste exemplo, é definida através da *WSDL* (Newcomer, 2002).

3.3 *Web Services* Semânticos

Os serviços *Web*, que há anos são a base das arquiteturas orientadas a serviços, começam a apresentar deficiências principalmente nas áreas de descrição, descoberta e composição de serviços. O problema está na falta de suporte semântico na linguagem de descrição de serviços *WSDL* e no mecanismo de armazenamento e pesquisa de serviços *UDDI*.

Com o propósito de integrar a *Web* Semântica aos serviços *Web*, foram desenvolvidas novas linguagens para descrição e composição de serviços apresentadas na **Figura 9**. Essa integração ainda está em curso, e as atuais extensões semânticas do *UDDI* (Naumenko et al., 2005), (Srinivasan et al., 2004) não garantem a compatibilidade e desempenho necessário.

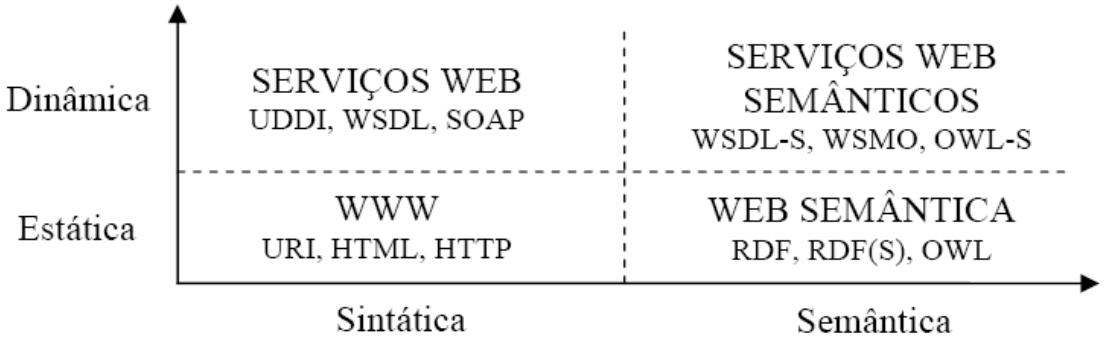


Figura 9 - Evolução das tecnologias *Web*.

3.4 Web Ontology Language for Services (OWL-S)

OWL-S é uma linguagem de serviços *Web* baseada na *OWL* que disponibiliza aos provedores de serviços *Web* um conjunto de linguagens de marcação construído para descrição das propriedades e capacidades de serviços *Web* de modo não ambíguo e interpretável por computadores. A *OWL-S* facilita a automatização das tarefas de serviços *Web*, incluindo automatização da descoberta, execução, composição e inter-operação, de serviços (Martin et al., 2007).

3.4.1 Service

O *OWL-S* pode ser subdividido em três partes, cada uma responsável por uma tarefa particular, que descrevem as seguintes características de um serviço *Web*: o que o serviço faz é apresentado no *Service Profile*, como o serviço trabalha é descrito pelo *Service Model*, e o acesso ao serviço é suportado pelo *Service Grounding* **Figura 10**.

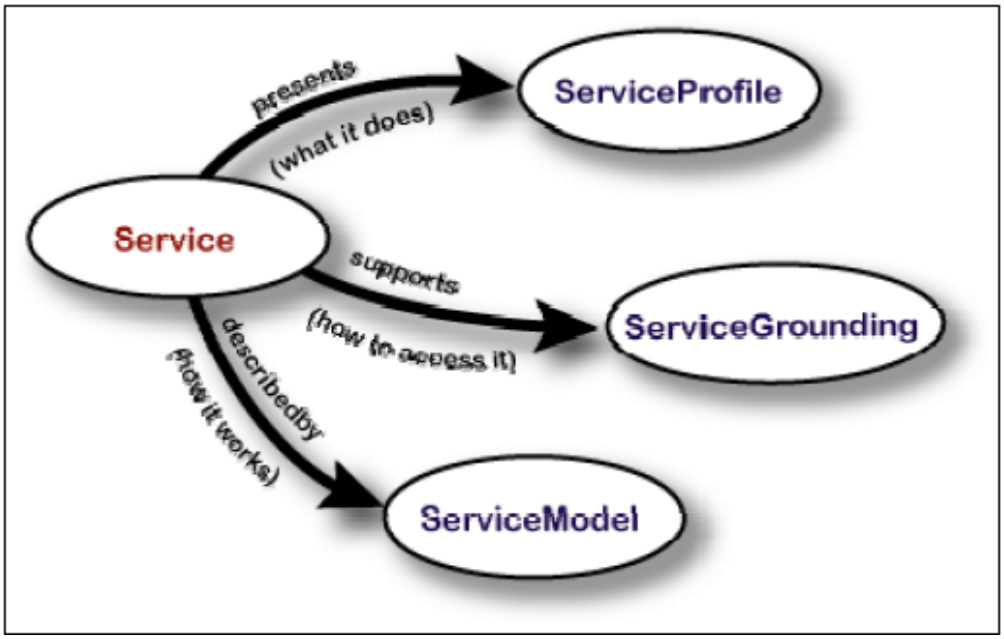


Figura 10 - Ontologia *Service* (kaul, 2006).

3.4.2 Service Profile

O *Service Profile* é uma descrição que efetua uma descrição de alto-nível do provedor de serviços e das funcionalidades do serviço. O *Service Profile* descreve as seguintes características: informações de contato de um determinado provedor/serviço (e.g., *serviceName*, *textDescription*, *contactInformation*); atributos para categorização do serviço oferecido (e.g., *serviceParameter*, *serviceCategory*); e as representações funcionais do serviço em forma de *Inputs Outputs Preconditions Effects* (*IOPEs*). Os *IOPEs* são descritos pelas propriedades *hasParameter*, *hasInput*, *hasOutput*, *hasPrecondition*, *hasEffect*.

Na **Figura 11**, são mostradas as classes e propriedades da ontologia que descreve o *Profile*.

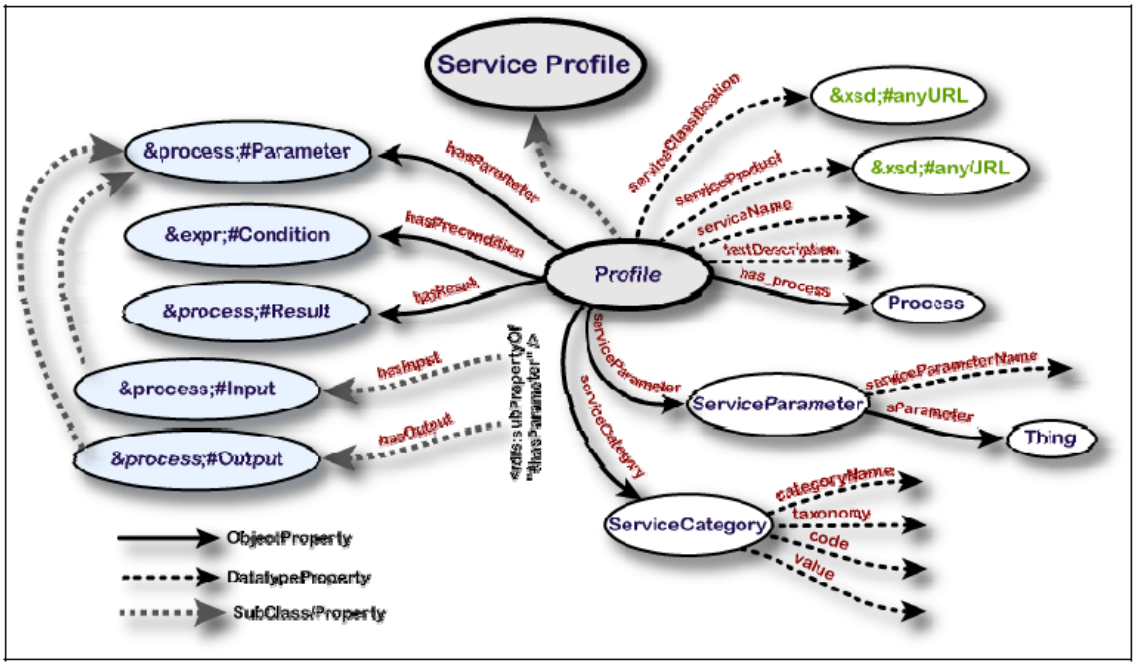


Figura 11 - Classes selecionadas e propriedades do *Profile* (Kaul, 2006).

Os três componentes são necessários para se usar um serviço *Web*: o requisitante que busca um serviço, o provedor que oferece um serviço, e os componentes de infra-estrutura que agem como registros para auxiliar o requisitante na localização do provedor de serviço apropriado. O papel de um *Service Profile* é ser publicado em um registro, porque ele provê as informações necessárias para a descoberta de um serviço.

O *Profile* também pode descrever os requisitos não funcionais do serviço, aspectos de qualidade de serviço (como a localização geográfica, categoria do serviço, segurança etc.) que podem ser usados por mecanismos de inferência ou algoritmos de busca semântica para que estes descubram os serviços com a melhor qualidade dentre aqueles que ofereçam funcionalidades semelhantes, durante o processo de busca de serviços semânticos. ‘

Através do *Service Profile,* é possível descrever as características do perfil de serviço, incluindo as condições necessárias para a sua execução. Essas características podem ser divididas em duas classes: as diretamente associadas ao processo de adaptação, representadas pelas Entradas e Saídas; e as de apoio, representadas pelas Precondições e Efeitos, que

auxiliam a decisão da política de adaptação em executar, ou não, um determinado serviço, conforme ilustra a **Figura 12**.

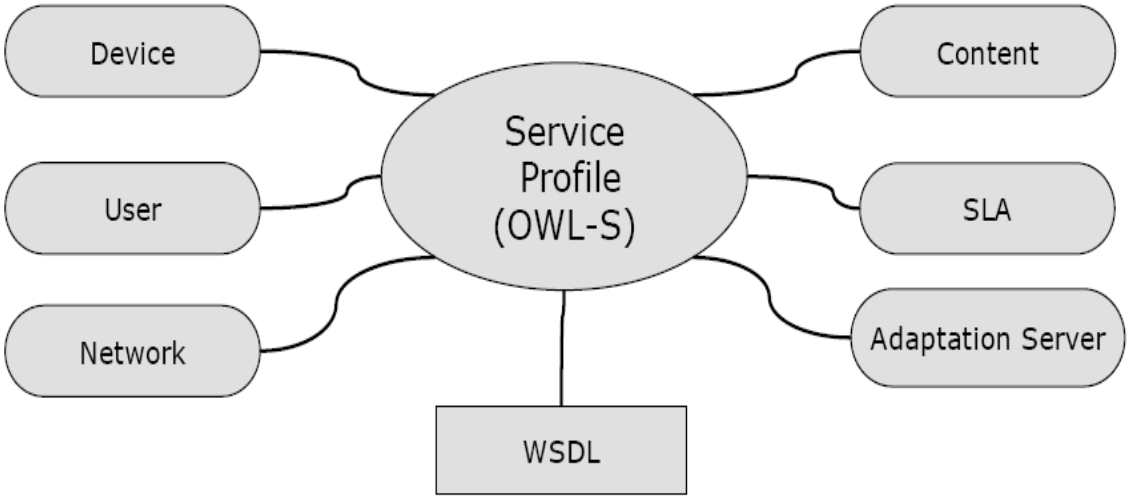


Figura 12 - Descrições de perfis e serviços que podem ser importadas pelo perfil de serviço.

3.4.3 Service Model

O *Service Model* é uma descrição do fluxo de trabalho definindo como o serviço opera. Uma subclasse do *Service Model* é o *ProcessModel*, onde os serviços são representados através de processos. Cada processo é visto como uma transformação de dados de um conjunto de entradas para um conjunto de saídas e como uma transição no contexto de um estado para outro, onde essa transição é descrita por pré-condições e efeitos. Ele especifica as formas de um agente interagir com um serviço (Martin et al., 2007).

Para poder executar esse processo, um agente necessita gerar um modelo de execução semanticamente equivalente, por meio da transformação do fluxo de dados e do fluxo de controle em conjunto com as condições associadas. Depois da instanciação desse modelo, o agente deve executá-lo passo a passo.

Um processo pode ser classificado em três classes:

- **Processo atômico (*AtomicProcess*):** pode ser executado diretamente, sem a necessidade de sub-processos;
- **Processo composto (*CompositeProcess*):** um processo que é composto de um processo atômico ou um conjunto de

processos e o qual pode ser decomposto pelas partes que o compõe utilizando um ou mais funções de construção (*Sequence, Concurrent, Split, Split+Join, Unordered, Choice, If-Then-Else, Repeat-Until, Repeat-While*);

- **Processo simples (*SimpleProcess*):** um processo abstrato que provê uma visão alternativa de um *AtomicProcess* ou uma abstração de um *CompositeProcess.*

A **Figura 13** apresenta uma visão detalhada da ontologia de processo:

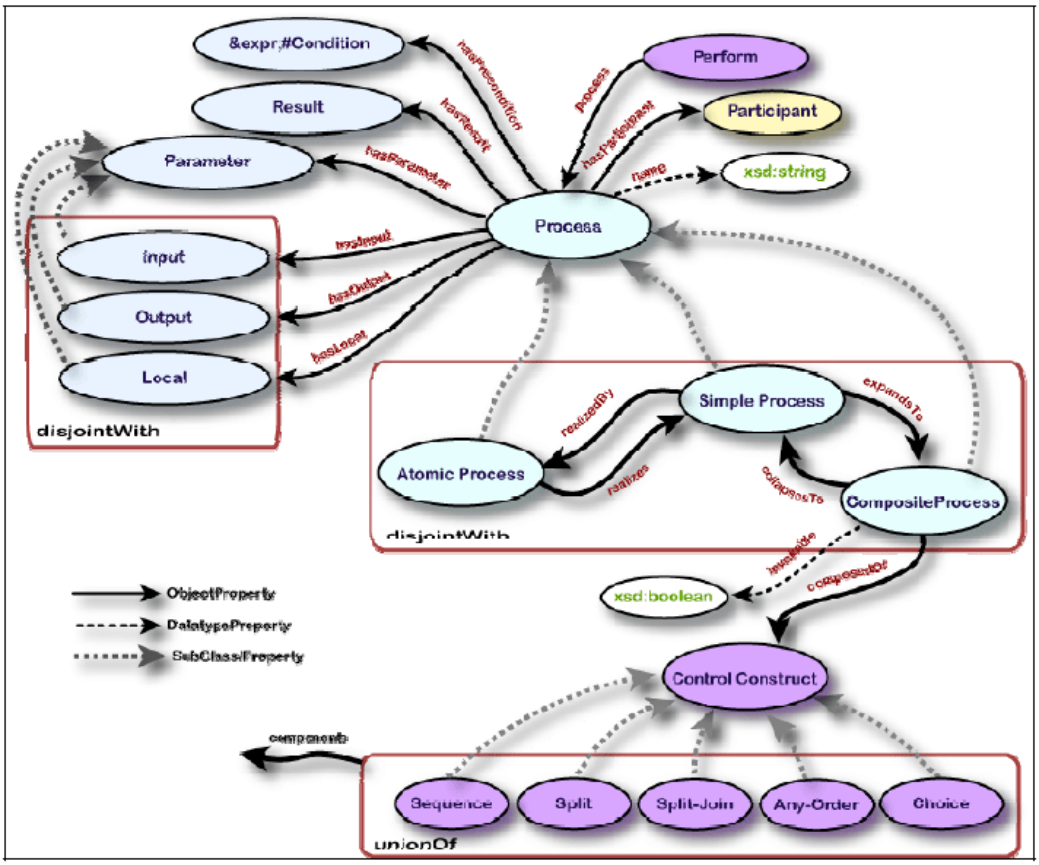


Figura 13 - Visão da Ontologia de Processo (kaul, 2006).

3.4.4 Service Grounding

O *Service Grounding* especifica os detalhes de como acessar o serviço. É a única parte de um serviço *OWL-S* em um nível concreto de especificação, considerando que o *Service Model* e o *Service Profile* são mais abstratos. Está altamente relacionado com *WSDL*: um processo atômico do *OWL-S* corresponde a uma operação do *WSDL*.

A **Figura 14** apresenta uma visão geral sobre o relacionamento entre o *Grounding* de um serviço web semântico e o *WSDL* do serviço concreto.

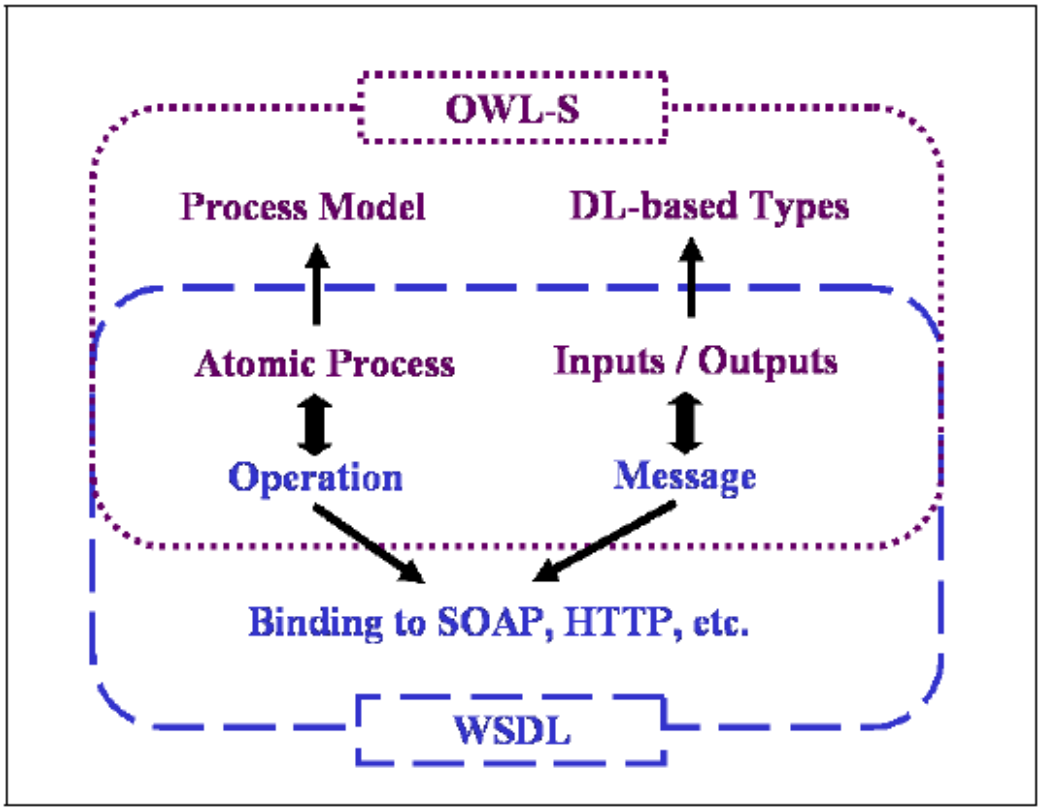


Figura 14 - Relacionamento entre *OWL-S Grounding* com *WSDL* (kaul, 2006).

3.5 A API "OWLS-UDDI matchmaker"

Esta API é fruto do trabalho de (Srinivasan et al., 2004) e possui como principais contribuições a extensão do mecanismo de registro de serviços do *UDDI*, ao integrá-lo com a tecnologia da *Web* Semântica, a fim de prover a descoberta de *Web Services* automática; e ao propor um *matchmaker* para realizar a descoberta de serviços com base nos conceitos definidos em ontologias de serviço. Para isso, os autores criaram o *OWLS-UDDI matchmaker* como pode ser visto na **Figura 15**.

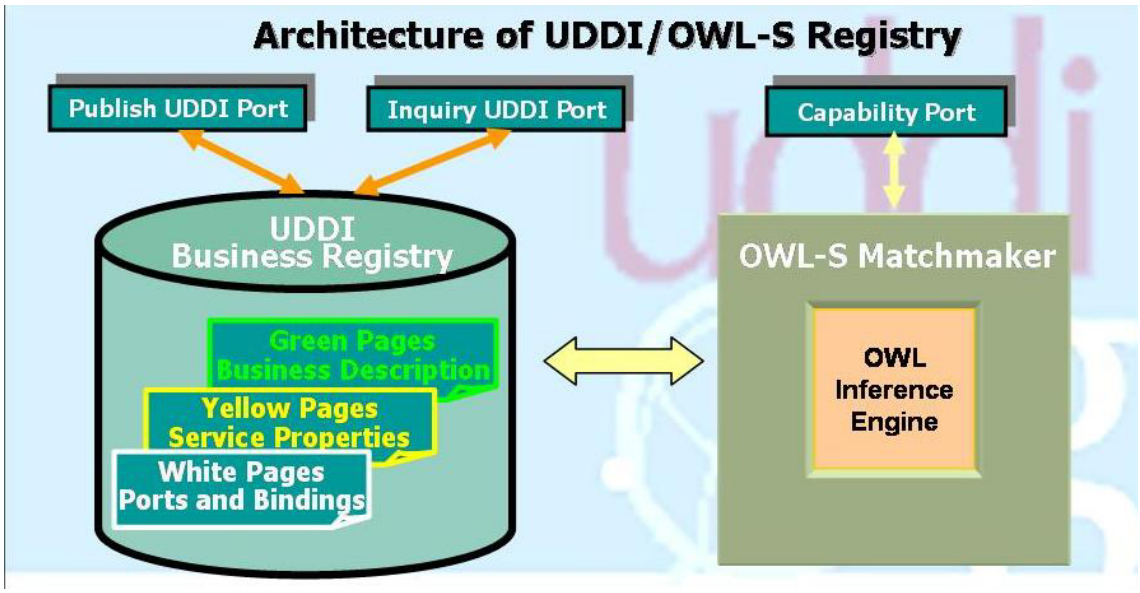


Figura 15 - Arquitetura do *OWL-S / UDDI Matchmaker* (Srinivasan et al., 2004).

Apesar de o *UDDI* ser a tecnologia padrão para armazenar a descrição dos serviços (*WSDL*) de um *Web Service*, ele possui duas importantes limitações no seu mecanismo de descoberta.

Os autores afirmam que: [...] "*A primeira limitação é o mecanismo de busca. Em UDDI, um Web Service pode descrever suas funcionalidades usando esquemas de classificação como NAISC, UNSPSC, etc. [...] Embora nós possamos descobrir Web Services usando os mecanismos de classificação, a busca obterá resultados não refinados, com alta precisão. A segunda desvantagem do UDDI é o uso do XML para descrever seu modelo de dados. UDDI garante a interoperabilidade sintática, mas falha ao descrever uma descrição semântica do seu conteúdo. Portanto, duas descrições XML idênticas podem ter significados muito diferentes, e vice versa.*".

A fim de solucionar as limitações do *UDDI*, os autores propuseram estender o mecanismo de registro, ao adicionar ontologias de serviço (*OWL-S*) ao *UDDI*. Eles adicionaram a ontologia do *Profile* ao mecanismo de registro. Ontologias de serviço, como *OWL-S*, tentam solucionar o problema da falta de descrição semântica que o mecanismo de registro tradicional possui.

Segundo os autores, assim como o *UDDI*, *OWL-S* também descreve *Web Services* através de esquemas de classificação. E ainda pode tirar

proveito da descrição da capacidade dos serviços (a capacidade de expressar as entradas, saídas, pré-condições e efeitos) que este possui. E afirmam, ainda, que esta descrição das capacidades do serviço irá superar as limitações do *UDDI* e prover melhores resultados nas buscas.

Neste trabalho, foi feito um mapeamento dos conceitos definidos na ontologia do *Profile* para os campos armazenados nos registros do *UDDI* (*UDDI*.org, 2008), mais especificamente o *TModel*, "conjunto de propriedades ilimitadas que podem ser associadas com uma especificação *Web Service*", foi estendido a fim de permitir o mapeamento dos conceitos definidos no *Profile* (Martin et al, 2007).

A **Figura 16** representa a correspondência entre os conceitos definidos no *Profile* e os respectivos campos no *TModel*.

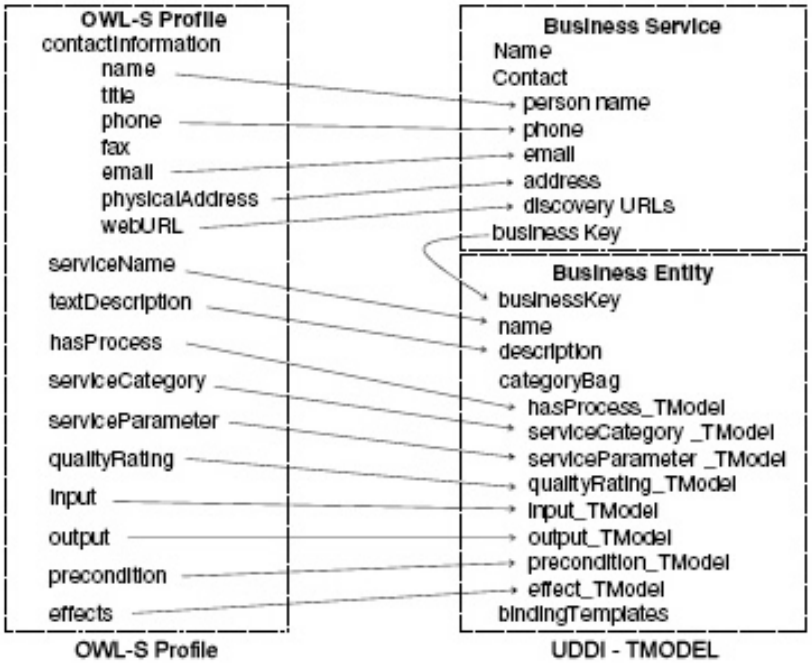


Figura 16 - Mapeamento *OWL-S Profile UDDI* - TMODEL (Srinivasan et al., 2004).

Ao mecanismo de registro também foi adicionado um *matchmaker*. Um *matchmaker* é um algoritmo baseado em regras lógicas, que utiliza mecanismo específico para inferir os conceitos e propriedades definidas nas ontologias, com a finalidade de descobrir novos conhecimentos.

O algoritmo de casamento (*matching*) utilizado pelo *matchmaker* funciona da seguinte forma: quando um usuário e/ou agente de *software* requisita um serviço ao *UDDI*, a requisição de busca é tratada pelo

matchmaker. Este por sua vez, procura por serviços cujos conceitos na ontologia de domínio são equivalentes aos passados através da requisição do usuário. Quando o algoritmo encontra o(s) serviço(s) que satisfazem a requisição, este(s) é/são, em um segundo momento, comparado(s) com as entradas da ontologia de domínio passadas pelo usuário. Caso algum serviço atenda à requisição do usuário, a(s) chave(s) que identifica(m) o(s) serviço(s) encontrado(s) é/são retornada(s) pelo *matchmaker*. Além disso, também são retornadas pelo *matchmaker* informações sobre o nível de casamento, além de informações sobre o mapeamento das entradas, saídas, pré-condições e efeitos (IOPES) da requisição, assim como das IOPES do serviço anunciado pelo *UDDI*. O algoritmo trabalha com quatro níveis de *matching*: exato, *plug in*, *subsume* e falha.

- **Exato:** O casamento é dito exato quando os conceitos definidos, tanto pela saída da requisição do usuário quanto o anotado na saída da ontologia de serviços, são iguais, ou quando o conceito definido na saída da requisição do usuário é uma subclasse do conceito definido na ontologia de serviços;
- ***Plug in*:** um relacionamento é dito ser do tipo *plug in* se o conceito definido pela saída da ontologia de serviço absorve, inclui o conceito definido pela saída da requisição do usuário, ou seja, o resultado desejado pode ser substituído pelo conceito definido pela ontologia de serviço (conceito mais geral). Este tipo de casamento é inferior ao definido pelo tipo exato;
- ***Subsume*:** É o relacionamento contrário ao definido pelo tipo *plug in*. Ocorre quando o conceito definido através da saída da requisição do usuário absorve o conceito definido na ontologia de serviços. Este casamento é inferior tanto ao do tipo exato quanto do tipo *plug in*, uma vez que, o serviço recuperado pode não atender de forma total a requisição do usuário, fazendo-se necessária, por exemplo, uma composição de serviços;
- **Falha:** Ocorre quando os conceitos definidos pela requisição do usuário não casam com nenhum conceito definido pelas ontologias de serviços dos anúncios armazenados pelo *UDDI*, nem mesmo podendo ser realizados casamentos do tipo *plug in* ou *subsume*.

Apesar de esta abordagem ter solucionado os problemas encontrados pelo mecanismo de registro tradicional, um dos pontos fracos é que tornou o tempo de publicação do serviço no *UDDI* significativamente maior (6 a 7 vezes em relação ao mecanismo tradicional).

Uma vez que deve carregar todos os conceitos da ontologia e validá-los através de um engenho de inferência, como o *Jena* (*Jena*, 2008), para certificar-se que estes não estejam inconsistentes. Contudo, os autores deram mais importância à fase de consulta ao *UDDI*, a qual se mostrou ter um tempo praticamente constante.

3.6 Considerações Finais

A proposta da *Web* Semântica se baseia em um conjunto de padrões que possibilita uma interpretação semântica de informações disponíveis na *Web*. Para expressar a semântica de um determinado conteúdo, são incorporadas informações que descrevem e relacionam os seus componentes, essas informações são baseadas em ontologias.

O uso de ontologias possibilita uma interpretação não ambígua da informação. Essa integração possibilita o uso de mecanismos de raciocínio (*reasoner*), facilitando a descoberta e composição de conhecimento.

No caso deste trabalho, a descrição semântica do contexto de entrega e dos serviços de adaptação disponíveis facilita o desenvolvimento de uma política de adaptação mais genérica e precisa. Essas características da *Web* Semântica começam a ser integradas na arquitetura de serviços *Web*. Arquitetura que possui uma grande adoção pelo mercado devido a sua independência de fornecedor, linguagem de programação, e plataforma. Essa integração está sendo desenvolvida através de novas linguagens como a *OWL-S* e a *WSDL-S*.

Nesta dissertação, foi adotada a *OWL-S* para descrever os serviços de adaptação. A *OWL-S* propicia uma rápida integração entre ontologias e os padrões de serviços *Web* (e.g., *WSDL*), além disso, integra uma linguagem de regras, a *SWRL*, que será utilizada pelo mecanismo de descoberta e composição de serviços.

Esta dissertação define a construção de uma arquitetura baseada em *serviços web* semânticos para a Computação Pervasiva, a fim de possibilitar a realização de uma busca semântica de serviços, conforme a necessidade do cliente. Para isso, utiliza a API *OWLS-UDDI matchmaker* (Srinivasan et al., 2004) para realizar uma parte da implementação da arquitetura proposta.

4 TRABALHOS RELACIONADOS À *WEB* SEMÂNTICA E COMPUTAÇÃO PERVASIVA

Este capítulo descreve alguns trabalhos desenvolvidos na área de computação ciente de contexto, apresentando as suas vantagens e desvantagens.

Atualmente, existem vários projetos que utilizam ontologias para a representação de contextos que refletem a situação do usuário no mundo real. Essas ontologias podem ser construídas utilizando a linguagem *OWL* e servirem aos propósitos da *Web* Semântica. Dessa forma, as instâncias dos conceitos podem referenciar uma pessoa, um dispositivo computacional ou mesmo um local.

4.1 SOUPA

Um projeto bastante conhecido nessa área é o *Standard Ontology for Ubiquitous and Pervasive Applications* (*SOUPA*) (Chen et al., 2004), que utiliza a *OWL* na criação de suas ontologias. O objetivo do projeto é definir ontologias para suportar aplicações destinadas à Computação Pervasiva. O desenvolvimento do *SOUPA* é conduzido por um conjunto de casos de uso. O vocabulário do *SOUPA* utiliza o vocabulário de algumas ontologias existentes e seu mérito está em providenciar aos desenvolvedores de aplicações uma ontologia que combina muitos vocabulários úteis de diferentes ontologias consensuais.

As ontologias que são referenciadas pelo *SOUPA* são: Friend-Of-AFriend (*FOAF*) (Brickley e Miller, 2006), *DAML-Time* (Pan e Hobbs, 2004), as ontologias sobre espaço da *OpenCYC* (Lenat e Guha, 1990), Regional Connection Calculus (*RCC*) (Randell et al., 1992), *COBRA-ONT* (Chen et al., 2004), *MoGATU BDI* (Perich, 2004) e a ontologia de políticas Rei (Kagal et al., 2003). Dentre as características de cada uma pode-se destacar:

• ***FOAF***: essa ontologia permite a expressão de informações pessoais e relacionamentos. É útil para a criação de sistemas de informação que suportam comunidades online. Aplicações de Computação Pervasiva usam essa informação para raciocinar sobre o perfil e conexões sociais da pessoa em relação aos seus vizinhos.

• ***DAML-Time*****:** os vocabulários dessa ontologia são desenvolvidos para expressar conceitos temporais e propriedades comuns a qualquer formalização de tempo. Aplicações de Computação Pervasiva podem usar essa ontologia para compartilhar representações comuns de tempo e para raciocinar sobre as ordens temporais de diferentes eventos.

• **Ontologias espaciais *OpenCYC* e *RCC***: as ontologias do *OpenCYC* definem um conjunto compreensivo de vocabulários para representação simbólica de espaço. A ontologia *RCC* consiste em vocabulários para expressar relações espaciais para raciocínio qualitativo de espaço. Em aplicações de Computação Pervasiva essas ontologias podem ser exploradas para descrever e raciocinar sobre contexto de localização.

• ***COBRA-ONT* e *MoGATU BDI*:** essas ontologias são destinadas a suportar a representação de conhecimento e inferências em ambientes de Computação Pervasiva. Enquanto o projeto do *COBRA-ONT* foca na modelagem do contexto em salas de reuniões inteligentes o projeto do *MoGATU BDI* foca na modelagem de crenças, desejos e intenções de usuários humanos e agentes.

• **Ontologia de política Rei**: a ontologia Rei define um conjunto de conceitos de direitos e deveres (e.g., direitos, proibições, obrigações e desobrigações) para especificar e inferir sobre regras de segurança e controle de acesso. Em ambientes de Computação Pervasiva os usuários podem usar essa ontologia de política para especificar regras de alto nível para garantir e revogar os direitos de acesso para/de diferentes serviços.

• **Ontologias *SOUPA***: consiste de dois conjuntos distintos, porém relacionados, de ontologias: *SOUPA Core* e *SOUPA Extension*. As ontologias do *SOUPA Core* pretendem definir os vocabulários

genéricos que são universais para diferentes aplicações em Computação Pervasiva. As ontologias do *SOUPA Extension* são extensões de ontologias básicas onde são definidos vocabulários adicionais para suportar tipos específicos de aplicações e providenciar exemplos para futuras extensões de ontologias.

4.2 SOCAM

O *Service-Oriented Context-aware Middleware* (*SOCAM*) é uma arquitetura para a construção de um serviço móvel ciente de contexto (Gu et al., 2004). O modelo do contexto é baseado em ontologia que provê um vocabulário para representar e compartilhar conhecimento de contexto em um domínio da computação onipresente. O projeto da ontologia do contexto é composto de dois níveis hierárquicos. Um nível contém ontologias individuais sobre vários subdomínios, por exemplo, o domínio de uma casa, um escritório, um veículo, etc. Um nível mais alto contém conceitos gerais sobre as outras ontologias, esse nível é chamado de ontologia generalizada ou ontologia de alto nível (Gu et al., 2004).

O domínio específico de uma ontologia pode ser dinamicamente re-alocado. Por exemplo, quando um usuário deixa sua casa para dirigir um carro, a ontologia do domínio da casa será trocada automaticamente pela ontologia do veículo. A ontologia é escrita em *OWL* (*W3C*, 2008).

A arquitetura *SOCAM* é composta pelos seguintes elementos **Figura 17**:

- **Provedores de contexto**: provêem uma abstração do sensoriamento de baixo nível. Cada provedor de contexto precisa ser registrado em um serviço de registro, o *SLS* para que outros usuários possam descobrir esses provedores. Os provedores de contexto podem ser externos ou internos. Os provedores de contexto externo obtêm contexto de fontes externas, por exemplo, um serviço de informação do tempo ou um serviço de localização *outdoor*. Os provedores de contexto interno obtêm contexto diretamente de sensores localizados em um subdomínio, como uma casa, por exemplo;

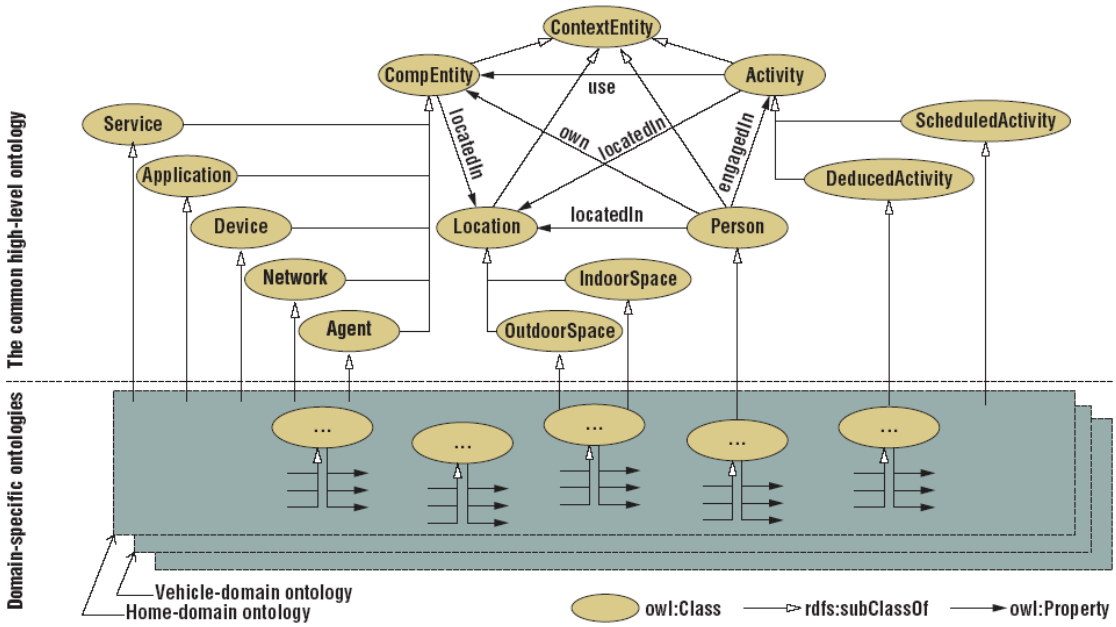


Figura 17 - Diagrama hierárquico de uma ontologia de contexto (Gu et al., 2004).

• **Serviço de Localização de Serviço (*SLS*):** permite usuários, agentes e aplicações descobrirem e localizarem diferentes provedores de contexto;

• **Interpretador de contexto:** provê contexto de alto nível através da interpretação de contexto de baixo nível. Dessa forma, o interpretador também é tratado como um provedor de contexto e pode ser registrado no *SLS*. O interpretador de contexto é dividido em *reasoner* e *Knowledge Base* (*KB*). O *reasoner* tem a função de prover contexto de alto nível baseado no contexto de baixo nível e detectar inconsistência e conflitos na base de conhecimento. *KB* provê um conjunto de API para que outros componentes possam consultar adicionar, apagar ou modificar conhecimento de contexto, além de conter a ontologia de um contexto de um subdomínio e suas instâncias. Essas instâncias podem ser especificadas pelos usuários, no caso de contexto definido, ou adquirido de vários provedores de contexto;

• **Serviços cientes de contexto**: são aplicações que fazem uso dos diferentes níveis da arquitetura *SOCAM*. Os desenvolvedores dessas aplicações podem predefinir regras e especificar que métodos devem ser invocados quando uma condição for verdadeira. Todas as regras são salvas em um arquivo e pré-carregadas no *reasoner*.

SOCAM foi projetado como um serviço de componentes independentes que podem ser distribuídos numa rede heterogênea e podem se comunicar entre si. Todos os componentes são implementados em *Java*. Para a comunicação entre os componentes distribuídos é usado *Java RMI*, que permite objetos distribuídos invocarem métodos de outros objetos remotos. A interação entre os componentes do *SOCAM* ocorre, resumidamente, da seguinte forma: um provedor de contexto pode adquirir dados sobre o contexto de vários sensores heterogêneos. Diferentes provedores de contexto registram seus serviços no *SLS*. As aplicações móveis ciente de contexto são capazes de localizar um provedor e obter dados sobre um determinado contexto. O interpretador de contexto e o serviço móvel ciente de contexto também podem ser registrados no *SLS*.

A arquitetura *SOCAM* segue uma arquitetura semelhante ao padrão *Web Service*, na qual os serviços são registrados em um diretório público e podem ser encontrados e utilizados por outros serviços. Porém, a arquitetura não é independente de linguagem de programação, pois os componentes trocam mensagens usando *Java RMI*, tornando difícil a integração entre servidores heterogêneos. Além disso, as regras de contexto devem ser carregadas previamente no sistema para que passem a funcionar.

4.3 COMPASS 2008

COMPASS 2008 (Weißenberg et al., 2004) é um protótipo de uma plataforma de integração para *Web Services* inteligentes personalizados para as Olimpíadas de 2008 em *Beijing*. A principal idéia desse projeto é, através de um dispositivo móvel, realizar '*push*' de ofertas significantes baseadas na situação atual e no perfil do usuário.

Todas as dimensões do contexto (localização, calendário, perfil, etc) são representadas através de ontologias. Com agregação dessas dimensões, o *COMPASS* 2008 define situações que são registradas no sistema. Por exemplo, através de uma notação em *F-Logic,* é possível registrar a seguinte situação:

```
WatchingCompetition : BeingAtEvent [
position ->> loc#Stadium;
```

localAction ->> act#AnyAction;
userState ->> cal#Leisure].

Este exemplo descreve a situação “assistindo a uma competição”. O usuário se encontra nessa situação se estiver localizado no estádio executando qualquer ação e na sua agenda aquele horário está marcado como hora de lazer. O *COMPASS* 2008 pode ser composto de várias ontologias, como mostrado no exemplo anterior, pois, ‘*loc’*, ‘*act’* e ‘*cal’* são *namespaces* para diferentes ontologias.

Definida a situação em que o usuário se encontra e o seu perfil, o sistema se encarrega de buscar serviços que se encaixam nesses parâmetros. O perfil é composto por interesses, preferências e dados pessoais do usuário. Interesses são informações estáticas que não se alteram com a mudança de contexto, por exemplo, se o usuário possui interesse por música clássica e obras de arte. As preferências podem depender da situação, por exemplo, um usuário em sua cidade pode preferir comida italiana quando vai a um restaurante, mas quando ele está viajando pode preferir experimentar a comida local.

Cada usuário pode manter seu perfil através de um conjunto de propriedades que o caracterizam. Além disso, há sensores, que obtêm informações do ambiente atual do usuário (localização e tempo). O resultado são instâncias de uma ontologia de alto nível que são usadas para implicitamente construir um “perfil de situação” que é semanticamente comparado com os perfis de todas as situações conhecidas pelo sistema, e implicitamente comparada com todos os perfis de serviços registrados.

A desvantagem é que ele não trata o relacionamento com outros usuários, ou seja, o sistema não monitora o contexto de contatos do cliente. Além disso, a busca por produtos e serviços é baseada apenas na categoria do produto e no perfil do usuário, não se importando com outras características do produto ou propriedades do serviço.

4.4 Nexus

Nexus (Dürr, 2004) é uma plataforma com o propósito de dar suporte a todos os tipos de aplicações cientes de contexto através de um modelo de

contexto global. Servidores de contexto podem ser integrados à plataforma para compartilhar e usufruir das informações providas pelos outros servidores de contexto. Esses servidores de contexto são chamados de modelos locais (ou modelos de contexto).

Os modelos locais contêm diferentes tipos de informações do contexto. Por exemplo, um modelo que representa as estradas, um modelo que representa as casas em uma cidade, um modelo que representa as pessoas, entre outros. No caso do modelo de pessoas, são usados sensores para manter os dados atualizados no modelo.

A plataforma *Nexus* é organizada como mostra a **Figura 18** a seguir.

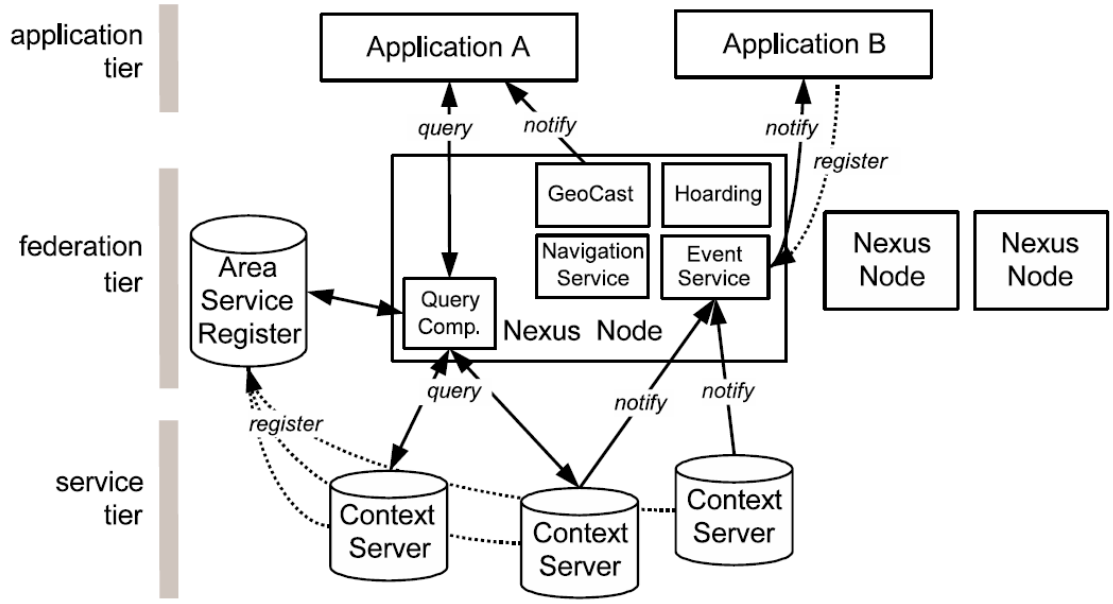


Figura 18 - Plataforma *Nexus* (Dürr, 2004).

Os servidores de contexto (*context servers*) presentes na camada de serviço (*service tier*) armazenam os modelos locais. Para serem integrados à plataforma, os serviços devem seguir certa *interface* descrita em *XML* e registrada em um serviço chamado *Area Service Register (ASR)* para que possam ser descobertos dinamicamente.

A camada de federação (*federation tier*) funciona como um mediador entre as aplicações e os servidores de contexto. Ele possui a mesma *interface* dos servidores de contexto, mas não armazenam modelos. Ele analisa a consulta da aplicação, determina os servidores de contexto que podem responder a consulta e distribui a consulta para esses serviços. O

nodo *Nexus* também possui a capacidade de adicionar serviços que possuem suas próprias *interface*s e que usam os modelos de contexto para processar informação e fornecê-la para as aplicações. A **Figura 18** mostra quatro tipos de serviços da plataforma *Nexus*:

- **O serviço de evento** (*event service*) monitora eventos espaciais e permite o processamento de predicados espaciais, tais como: “monitorar quando eu estiver próximo à dois amigos”;
- **O serviço de navegação** (*navigation service*) calcula a rota usando os dados disponíveis nos modelos locais;
- **O *geocast*** é responsável por enviar mensagens para todos os usuários em uma determinada área geográfica;
- **O serviço *Hoarding*** é responsável pelo processo de desconexão da aplicação de uma área de comunicação, ou seja, se uma aplicação está prestes a sair de uma área de comunicação de rede sem fio, o serviço Hoarding transfere as informações do contexto necessárias para essas aplicações com antecedência.

No topo da arquitetura estão as aplicações cientes de contexto que podem usar a plataforma de três formas diferentes:

- As aplicações podem enviar consultas e obter informações sobre o ambiente ao redor, como por exemplo, *PoI*, amigos próximos, etc;
- As aplicações podem registrar um evento para receber uma notificação quando um certo estado do mundo ocorrer;
- As aplicações podem usar os serviços do nodo *Nexus*, como os serviços de navegação, para enriquecer as funcionalidades da aplicação.

A plataforma *Nexus* possui uma arquitetura semelhante ao padrão *Web Service* (Alonso et al., 2004). Os servidores de contexto funcionariam como provedores de serviços que são registrados em um serviço de diretório; o *ASR* funcionaria como um serviço de diretórios semelhante ao *UDDI* no padrão *Web Service*. Porém, na plataforma *Nexus*, a descoberta de

serviços é realizada pelo nodo *Nexus* na camada de federação e não pelas aplicações como no padrão *Web Service*.

A plataforma *Nexus* monitora o espaço físico, ou seja, o contexto do ambiente. O *Nexus* é capaz de transmitir anúncios para vários usuários em uma determinada área, mas não faz busca automática de produtos ou serviços de interesse do usuário. Além disso, a plataforma não é capaz de deduzir informação a partir dos dados do contexto.

4.5 FieldMap

FieldMap (Field Map, 2007) é uma ferramenta escrita em *Java* para mostrar mapas e permitir anexar comentários. Ela foi construída para ser usada em *PDAs*, *desktops* e *laptops*. O usuário pode adicionar novos pontos de interesse, desenhar no mapa, marcar com pontos e polígonos e adicionar comentários escritos ou por voz para ajudar a recordar sobre informações do ambiente.

Pode-se perceber que esta ferramenta não faz qualquer análise do contexto do usuário. Apenas a localização e o caminho percorrido pelo usuário são mostrados no mapa para que ele possa se orientar. Essa ferramenta é mais utilizada na área de pesquisa na qual o ambiente está sendo estudado, como, por exemplo, na área de arqueologia.

4.6 Comparação entre as Ferramentas

Diferentes propostas para infra-estruturas de suporte ao contexto podem ser encontradas na literatura. Essas propostas resultam da diversidade de informações contextuais e de um espectro cada vez mais amplo para utilização de aplicações sensíveis a contexto (Mostéfaoui et al. 2004).

Foi realizado um estudo comparativo entre as infra-estruturas existentes atualmente. Neste estudo foram levadas em conta as seguintes características:

1. **Monitoração em tempo real**: O sistema deve ser capaz de monitorar as mudanças de contexto do usuário em tempo real.
2. **Monitoração de vários tipos de contexto**: O sistema deve ser capaz de comparar vários tipos de contexto para tomar decisão.

3. **Orientado a serviço**: as funcionalidades do sistema devem ser implementadas através de *Web Services* para facilitar a inclusão de novos serviços.
4. **Sistema SIG**: O sistema deverá gerenciar a localização do usuário.
5. **Análise do perfil do usuário para o fornecimento de mapas personalizados**: apresentar mapas conforme as preferências contidas no perfil do usuário.
6. **Anúncio de produtos ou serviços**: O sistema deverá ser capaz de permitir o registro de produtos e serviços.
7. **Execução de serviços**: o sistema deverá permitir a execução automática de serviços atômicos.

A **Tabela 2** mostra quais infra-estruturas possuem as características citadas anteriormente. Os campos marcados com 'X' indicam que a ferramenta possui a determinada funcionalidade. Os campos sem marcação indicam que a ferramenta não possui a funcionalidade especificada.

Características / Sistemas	1	2	3	4	5	6	7
SOCAM	X	X	X				
SOUPA	X	X					
COMPASS2008	X	Apenas localização e perfil	X	X	X	Apenas por tipo do produto	
Nexus	X	Apenas a localização		X			
FieldMap		Apenas a localização		X			

Tabela 2- Tabela de comparação das características das infra-estruturas estudadas.

4.7 Considerações Finais

Neste capítulo, foram descritos alguns sistemas cientes de contexto, como funcionam, suas principais vantagens e desvantagens. Por fim, foram descritas algumas comparações entre as funcionalidades dos sistemas apresentados e as características da arquitetura proposta, definindo pontos

importantes para um sistema ciente de contexto, capaz de monitorar o contexto dos usuários e realizar pesquisas por serviços dinamicamente.

.

5 SOMM – UMA ARQUITETURA PARA APLICAÇÕES CIENTES DE CONTEXTO

Este capítulo discorre sobre a modelagem da arquitetura do sistema SOMM (*Souto Onthology Matching Model*), contendo discussões a respeito dos principais elementos da arquitetura, composta por tecnologias da *Web* Semântica para realizar a descoberta e invocação automática de serviços web semânticos, descritos nas linguagens *OWL* e *OWL-S*, para manipular ontologias de domínio e fazer a anotação semântica dos serviços web e assim poder oferecer suporte a aplicações cientes de contexto para a Computação Pervasiva. Primeiramente, será abordada a metodologia de obtenção da arquitetura proposta.

5.1 Metodologia Utilizada

Segundo (Pressman, 2006), o projeto arquitetural representa a estrutura dos componentes de dados e programas que são necessários para construir um sistema baseado em computador. Ele considera o estilo arquitetural que o sistema vai adotar, a estrutura e as propriedades dos componentes que constituem o sistema e os inter-relacionamentos que ocorrem entre todos os componentes arquiteturais.

A representação da arquitetura permite ao engenheiro de *software*:

a) Analisar a efetividade do projeto em satisfazer os requisitos declarados;
b) Considerar alternativas arquiteturais em um estágio em que fazer modificações de projeto é ainda relativamente fácil;
c) Reduzir os riscos associados à construção de *software*.

A modelagem de uma arquitetura se faz necessária nesse projeto, por causa do grande número de requisitos existentes, e que não poderão ser implementados em uma única etapa dado o tempo de realização do trabalho. Outro fator é poder facilitar implementações futuras e garantir interoperabilidade e reuso. Em função desse quesito, a arquitetura proposta utiliza a abordagem de uma arquitetura orientada a serviço.

A idéia principal do projeto é a disponibilização de serviços para o usuário de forma transparente, onde esses serviços são requisitados através de um dispositivo móvel. O dispositivo móvel possui uma grande mobilidade, limitações de recursos como tamanho da tela, pouca capacidade de processamento e dificuldades de manipulação de entrada e saída de dados. Em função desses fatores, esses serviços têm que ser elaborados de maneira que o usuário não necessite de muito esforço para acessá-los.

Atender todos os requisitos necessários em um tempo curto é praticamente inviável. Em função disso, foi feito o levantamento de todos os requisitos necessários e manteve-se o foco na implementação do sistema de anotação, busca e execução de serviços semânticos **Figura 19**. Na próxima seção, são relatados os requisitos funcionais e não-funcionais da arquitetura SOMM.

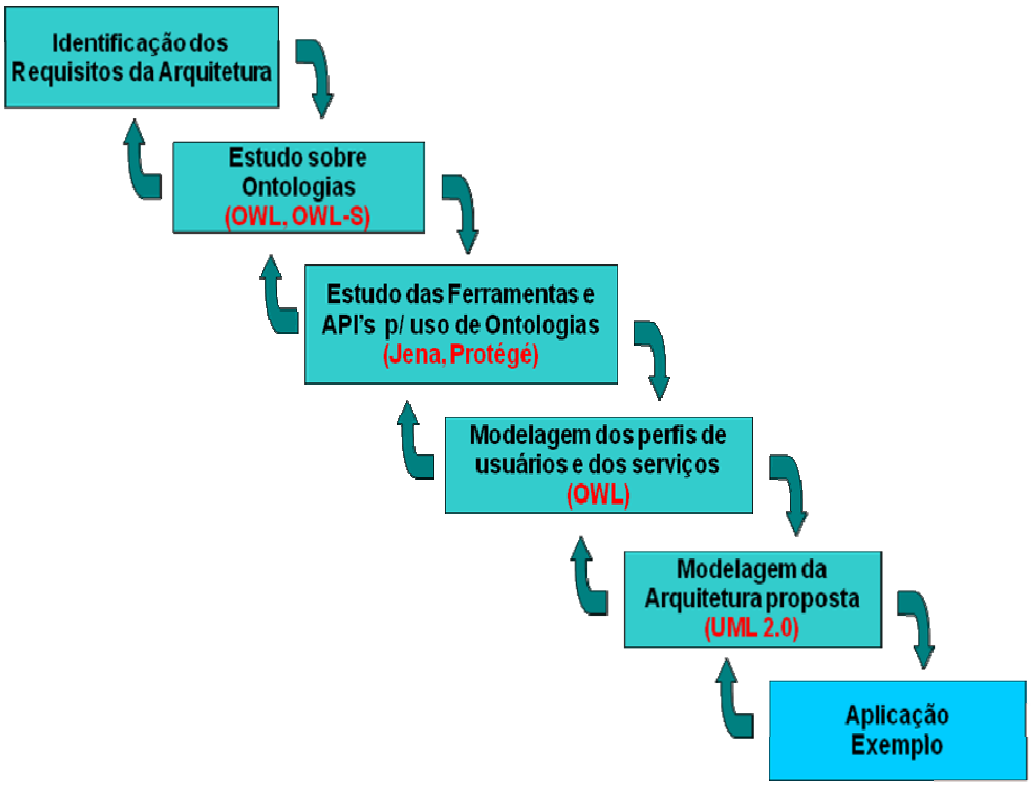


Figura 19 - Passos para execução do trabalho.

5.2 Requisitos Funcionais da Arquitetura

a) Funções existentes da página principal

O sistema deverá conter um controle de permissões no qual o usuário previamente cadastrado ao entrar no sistema visualizará as seguintes opções: inserir manualmente a sua localização / buscar serviços / anunciar serviços / mostrar mapa / editar perfil / buscar pessoas conhecidas / configurações / ajuda.

b) Armazenar o contexto do usuário e dos serviços

O SOMM deve ser capaz de armazenar o contexto do usuário e dos serviços. Atualmente, as informações de contexto do usuário que são monitoradas são: localização e interesse de serviços. O usuário é capaz de monitorar o próprio contexto.

c) Registrar anúncio do serviço

O SOMM deve incluir ou excluir serviços no *UDDI* semântico. Para registrar o mesmo, este serviço deve estar no formato *OWL-S*, pois o *matchmaker* possui um *parser* que verifica e valida à ontologia e com base no *Profile* registra o serviço na base de dados.

d) Descobrir Serviços

O SOMM deve buscar serviços semânticos no *UDDI* semântico, com base nos valores de entrada e saído passados pelos pedidos efetuados pelo sistema ou pelo usuário.

e) Executar Serviços semânticos

O SOMM deve possibilitar a execução dos serviços atômicos encontrados pelo serviço de busca.

f) Mapa personalizado

Além de o mapa possuir os pontos de interesse personalizados de acordo com as preferências do usuário, ele apresenta os contatos próximos e informações de serviços desejados.

5.3 Requisitos Não Funcionais da Arquitetura

a) Arquitetura Orientada a Serviço

As funcionalidades do SOMM devem ser implementadas através de *Web Services*. Os *Web Services* fornecem uma maneira estruturada de formatar dados, de tratar transações e um padrão para descrever o que o serviço faz e como torná-lo acessível a outros serviços. Um *Web Service* é uma *interface* que descreve uma coleção de operações acessíveis via rede através de troca de mensagens padronizadas em *XML*, que é a linguagem padrão de comunicação na *Web*. A *interface* esconde a implementação do serviço, permitindo que seja usado independentemente da plataforma, favorecendo a comunicação entre sistemas heterogêneos. *Web Services* também auxiliam no processo de desenvolvimento de *software*, pois permitem que funcionalidades distintas sejam desenvolvidas por equipes diferentes, agilizando o desenvolvimento do sistema e permitindo que os serviços sejam reaproveitados em várias aplicações (Alonso et al., 2004).

b) Resposta em tempo Hábil

O tempo de resposta rápido é fundamental em sistemas móveis, pois o usuário realiza uma busca e necessita obter uma resposta o mais breve possível.

c) Utilizar ontologias para armazenar o contexto

O SOMM deverá ser capaz de armazenar serviços anotados através de ontologias de serviço. Para tanto, considera-se que as ontologias de serviços sejam expressas em *OWL* e *OWL-S*, que fazem parte da recomendação *W3C* e são amplamente aceitas e utilizadas por parte da comunidade da *Web* Semântica (*W3C*, 2008).

d) Utilizar uma linguagem Orientada a Objetos

O SOMM será implementado utilizando uma linguagem de programação baseada no paradigma Orientado a Objetos, no caso *Java*, pelo fato da mesma ser multi-plataforma, o que tornará a arquitetura compatível com a maioria das plataformas computacionais atuais.

Com bases nesses requisitos levantados, as seções seguintes descreverão como a arquitetura foi modelada para atender os dados requisitos previamente expostos.

5.4 Os Atores

Os atores envolvidos diretamente na utilização da arquitetura SOMM podem ser classificados da seguinte forma:

- **Cliente**: deverá registrar o seu perfil com informações sobre seus interesses, entrará com as requisições da consulta (parâmetros de entrada e saída) com base em uma ontologia de domínio. A entrada dos dados será realizada através de uma *interface* web ou através de uma aplicação implementada em *JAVA ME*.
- **Provedor de serviços**: será o encarregado de realizar a anotação semântica dos serviços, publicar o serviço semântico através do *Web Service* de registro, e será encarregado pelo cadastramento dos mapas para serem disponibilizados ao cliente.

5.5 Diagrama de Casos de Uso

Uma visão geral sobre os casos de uso do SOMM é apresentada a seguir, através da **Figura 20**.

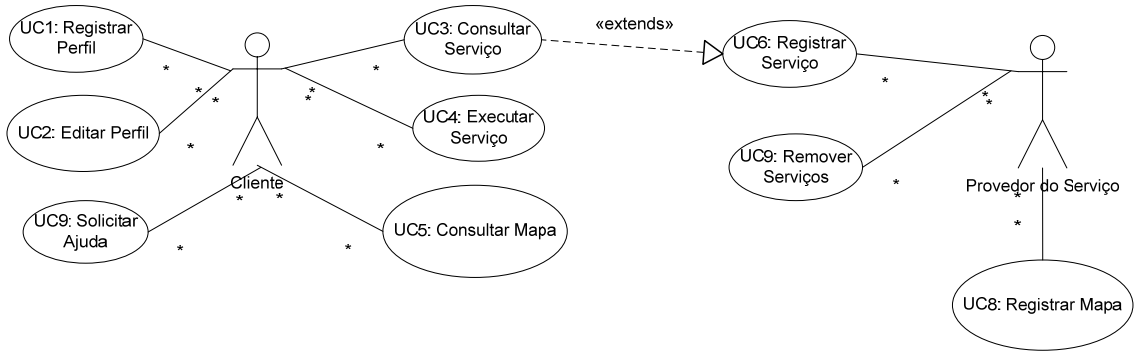


Figura 20 - Diagrama de Casos de Uso.

5.6 Descrição dos Casos de Uso

A descrição do diagrama de casos de uso demonstrado pela Figura_20 pode ser observada nas tabelas a seguir.

Tabela 3 - UC1: Registrar Perfil.

Descrição resumida: permite ao usuário registrar o perfil do usuário na base de dados.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Configurações do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Configurações;

b. o usuário seleciona a opção Registrar Perfil;

c. o sistema exibe o formulário com os campos para preenchimento;

d. o usuário preenche o formulário com informações sobre suas preferências e interesses.

Requisitos especiais: não se aplica.

Pré-condições: possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 4 - UC2: Editar Perfil.

Descrição resumida: permite ao usuário editar as informações sobre seu perfil.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Configurações do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Configurações;

b. o usuário seleciona a opção Editar Perfil;

c. o sistema exibe o formulário com os campos para preenchimento;

d. o usuário edita as informações.

Requisitos especiais: não se aplica.

Pré-condições:possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 5 - UC3: Consultar Serviço.

Descrição resumida: permite ao usuário pesquisar por serviços.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Consultar Serviço do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Consultar Serviço;

b. o sistema exibe o formulário com os campos para preenchimento;

c. o usuário preenche quais serão as informações de entrada e saída sobre o serviço que ela deseja consultar;

d. o sistema retorna o resultado com base nos dados passados pelo usuário.

Requisitos especiais: não se aplica.

Pré-condições:possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 6 - UC4: Executar Serviço.

Descrição resumida: permite ao usuário executar o serviço semântico com base nos conceitos de entrada e valores passados na requisição.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Consultar Serviço do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Consultar Serviço;

b. o sistema exibe o formulário com os campos para preenchimento;

c. o usuário preenche quais serão as informações de entrada e saída sobre o serviço que ela deseja executar;

d. o sistema retorna o resultado com base nos dados passados pelo usuário;

e. o usuário seleciona o serviço que ele deseja executar;

f. o sistema executa o serviço;

g. o usuário digita o valor de entrada;

h. o sistema retorna a resposta com base no valor de entrada.

Requisitos especiais: não se aplica.

Pré-condições:possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 7 - UC5: Consultar Mapas.

Descrição resumida: permite ao usuário visualizar um mapa.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Consultar Mapas do formulário principal.

Fluxo básico:

a. a. o usuário seleciona a opção Mapas;

b. o usuário seleciona a opção Consultar Mapas;

c. o sistema exibe o formulário com os campos para preenchimento;

d. o usuário digita as informações sobre a localização que ele deseja recuperar;

e. o sistema retorna o mapa com base nas informações que foram passadas pelo usuário.

Requisitos especiais: não se aplica.

Pré-condições: possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 8 - UC6: Registrar Serviço.

Descrição resumida: permite ao usuário registrar um novo serviço a base de dados.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Registrar Serviço do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Registrar Serviço;

b. o sistema exibe o formulário com os campos para preenchimento;

c. o usuário insere o endereço do arquivo que contêm a descrição do serviço no formato ".owl";

d. o usuário clica em registrar o serviço;

e. o sistema armazena os dados do serviço na base de dados.

Requisitos especiais: não se aplica.

Pré-condições: possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 9 - UC7 Remover Serviço.

Descrição resumida: permite ao usuário remover um serviço da base de dados.

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Remover Serviço do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Remover Serviço;

b. o sistema exibe o formulário com os campos para preenchimento;

c. o usuário insere o endereço do arquivo que contêm a descrição do serviço no formato ".owl";

d. o usuário clica em remover o serviço;

e. o sistema atualiza os dados dos serviços na base de dados.

Requisitos especiais: não se aplica.

Pré-condições: possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

Tabela 10 - UC8: Registrar Mapas.

Descrição resumida: permite ao usuário adicionar mapas no sistema

Fluxo de eventos: o caso de uso começa quando o usuário seleciona a opção Mapas do formulário principal.

Fluxo básico:

a. o usuário seleciona a opção Mapas;

b. o usuário seleciona a opção Adicionar Mapas;

c. o sistema exibe o formulário com os campos para preenchimento;

d. o usuário preenche o endereço de um WebService de localização de mapas.

Requisitos especiais: não se aplica.

Pré-condições: possuir uma conexão a rede com acesso a Internet e entrar no sistema.

Pós-condições: não se aplica.

Tabela 11 - UC9: Solicitar Ajuda.

Representa a ajuda no sistema.

Descrição resumida: permite ao usuário visualizar as informações de ajuda sobre o sistema.

Fluxo de eventos: não se aplica.

Fluxo básico:

a. o usuário seleciona a opção Ajuda;

b. o sistema exibe as informações sobre o funcionamento das funções do sistema SOMM.

Requisitos especiais: não se aplica.

Pré-condições: possuir uma conexão a rede com acesso a Internet.

Pós-condições: não se aplica.

5.7 Diagrama de Componentes

Uma visão geral sobre o relacionamento entre as classes do SOMM é representada pela **Figura 21.** Maiores detalhes são apresentados na subseção 5.7.1.

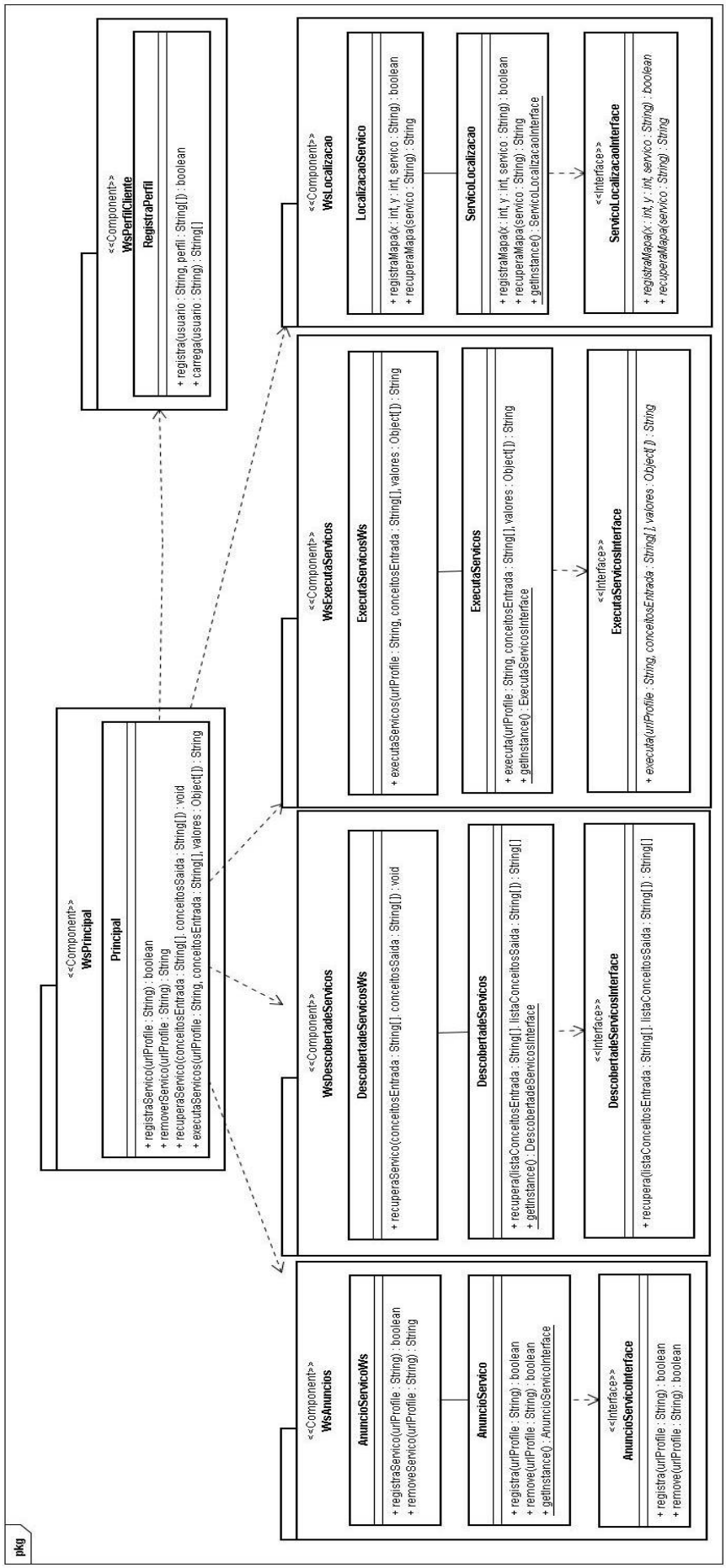


Figura 21 - Diagrama de Componentes do SOMM.

5.7.1 Classes

As principais classes da arquitetura são as classes: WsAnuncios, WsDescobertadeServicos e WsExecutaServiços, WsPrincipal, WsLocalizacao. Estas classes implementam os métodos definidos pelas suas respectivas *interface*s: WsAnunciosInterface, WsDescobertadeServicosInterface e WsExecutaServiçosInterface. Esses métodos realizam, respectivamente, as funcionalidades referentes aos serviços de registro, descoberta e execução.

É importante destacar que o relacionamento entre essas classes demonstra ser muito simples, em virtude das mesmas fornecerem métodos de acesso a *API´s* desenvolvidas por terceiros. Dessa forma, toda a complexidade poderá ser abstraída do desenvolvedor.

5.8 A Arquitetura do SOMM

A arquitetura do SOMM foi projetada e implementada utilizando a metodologia de construção de Arquiteturas Orientadas a Serviços (*SOA*) **Figura 22**. Dessa forma, será possível fazer inclusões de novas funcionalidades através da construção de novos *Web Services* que por sua vez podem ser implementados em qualquer linguagem de programação, não exigindo do desenvolvedor conhecimento do projeto como um todo.

Os *Web Services* que compõem a arquitetura são: Cliente (apresentação), Aplicação (negócio) e Base de Dados (persistência).

Uma visão geral da arquitetura do SOMM pode ser observada a partir da **Figura 23**, que mostra a aplicação dividida em serviços com os seus respectivos componentes.

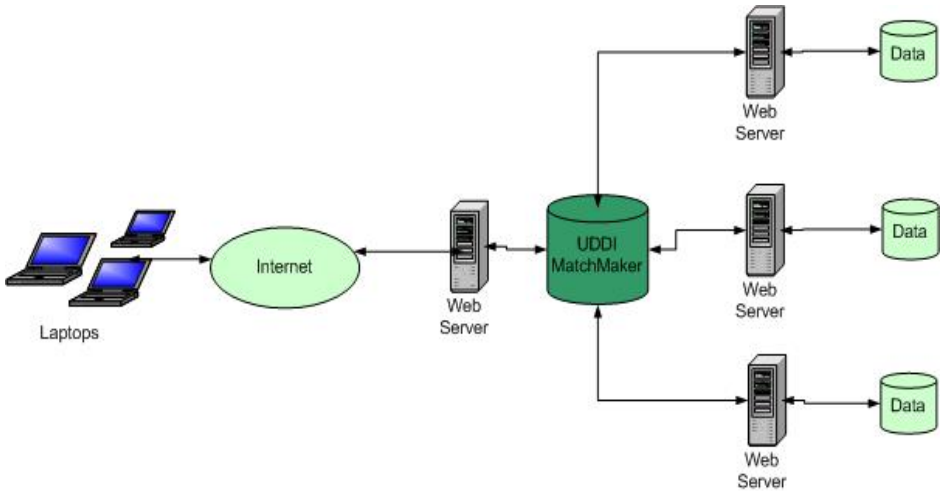


Figura 22 - Diagrama de Distribuição do UDDI Matchmaker.

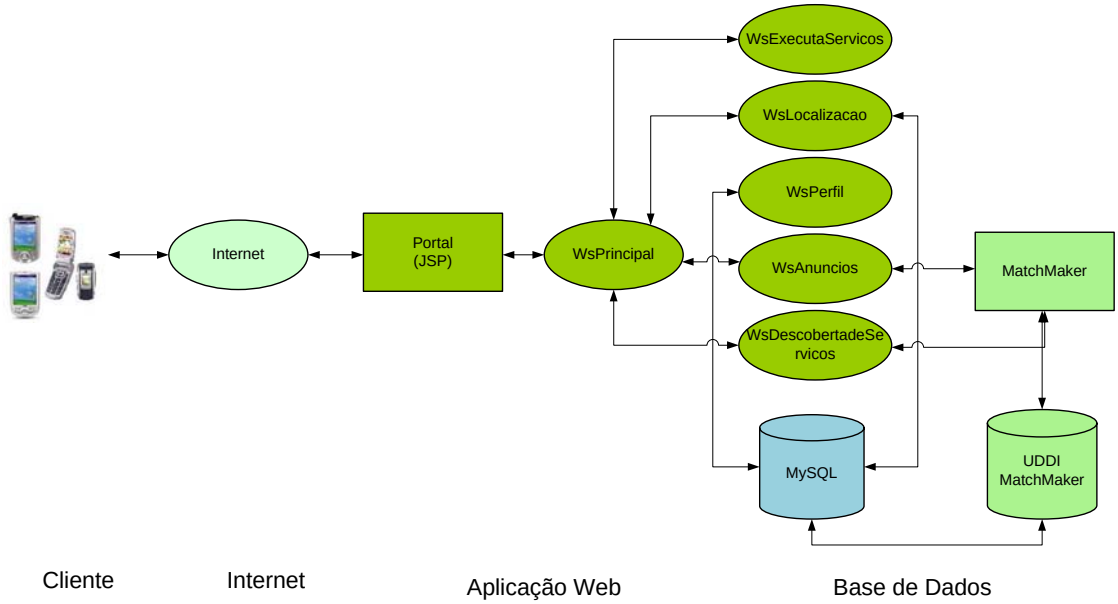


Figura 23 - Diagrama de Distribuição do SOMM.

5.8.1 Componentes

Foi escolhida a linguagem Java para a implementação dos *Web Services*, por ser uma linguagem que oferece diversas facilidades para a construção desses serviços e por ser a mesma linguagem utilizada nas API's utilizadas no projeto. Logo abaixo, é feita uma descrição resumida das camadas e componentes:

Clientes - Os serviços apresentados na camada *Web Services* podem ser acessados por um *browser* (através de uma aplicação *Web*) ou por um dispositivo móvel contendo um pacote *JAVA ME Web Services*. O

dispositivo móvel acessa diretamente os serviços como um cliente *Web Service*.

Aplicação WEB - Este nível contém as aplicações *Web* que acessam os serviços e processam as informações para que os usuários possam usufruir dos *Web Services* através de um *browser*. O principal objetivo dessa camada é permitir que os usuários possam acessar os serviços através de um PC e não apenas de um dispositivo móvel. Além disso, usando um *browser*, o usuário poderá executar tarefas que seriam inadequadas para um dispositivo móvel, como por exemplo, efetuar o cadastro de um novo serviço, devido a grande quantidade de dados que ele terá de digitar. Logo abaixo é apresentado um resumo das camadas, e na seção 5.8.4 é feito um detalhamento de cada uma delas:

a) **WsPrincipal -** é o principal serviço da arquitetura, responsável por gerenciar os usuários e efetuar o acesso aos outros serviços da arquitetura.
b) **WsAnuncios e WsDescobertadeServiços -** Estes *Web Services* gerenciam o anúncio e a busca de serviços oferecidos pelos usuários.
c) **WsPerfil -** Ele é responsável por receber e gerenciar informações de contexto. Através desse serviço, o usuário registra o seu perfil, adiciona regras que são ativadas de acordo com o contexto e atualiza sua posição, através de um dispositivo móvel equipado com *GPS* ou manualmente. Os dados capturados pelos sensores são transmitidos para este serviço, que por sua vez se encarrega de atualizar a ontologia e gerenciar as ações que devem ser disparadas.
d) **WsExecutaServiços** – Responsável por fazer a execução dos serviços encontrados.
e) **WsLocalização -** Este serviço é responsável por apresentar mapas em um dispositivo móvel tanto para dispositivos móveis quanto para uma aplicação *Web* executando em um *PC.* Possui ainda capacidade de fornecer mapas de acordo com o perfil do usuário.

5.8.2 Modelagem Conceitual

Para realizar inferências sobre os perfis do usuário e a descrição dos serviços é necessário armazenar as informações através de ontologias bem definidas. Para o caso da arquitetura, foram definidos as seguintes modelos:

- **Modelo conceitual do perfil do Usuário** - trata as informações do usuário, como por exemplo, status, interesses e localização;
    - **Modelagem Conceitual do perfil do usuário**

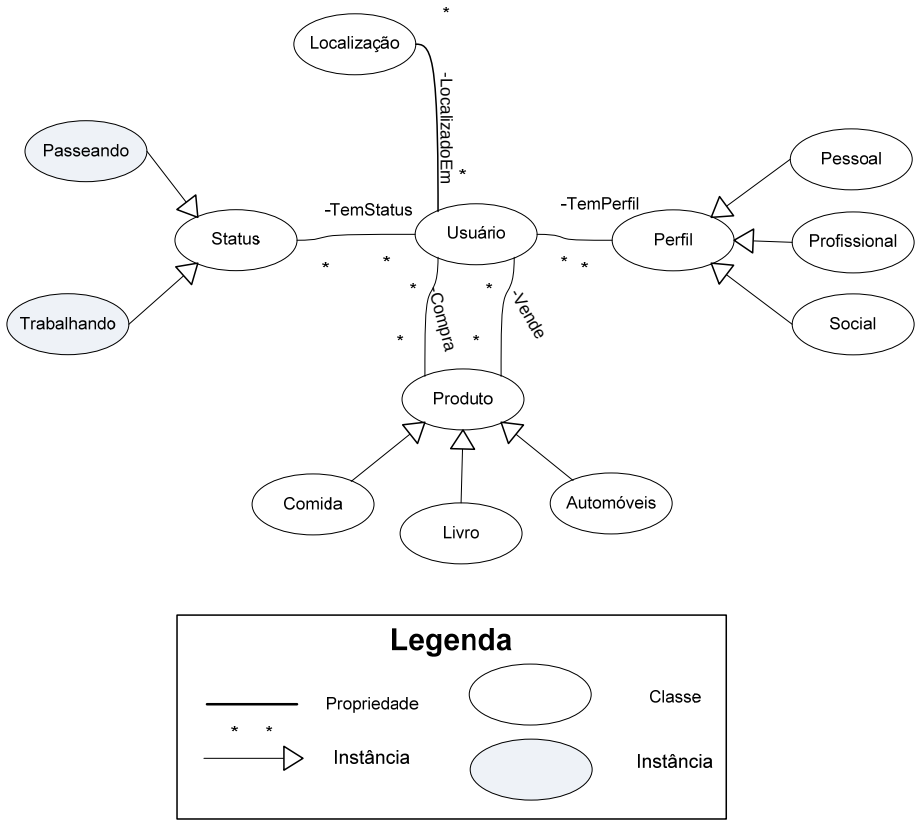


Figura 24 – Modelo conceitual do Usuário.

- **Modelo conceitual do Serviço -** trata as informações sobre os serviços existentes como o nome do serviço, endereço, telefone, tipo, valor e localização;
- **Modelo conceitual do Domínio -** trata de questões ligadas à estrutura hierárquica do domínio onde a arquitetura será utilizada.

5.8.3 A Base de Dados

A camada de persistência é constituída pelo serviço de diretório, implementado através do *jUDDI*, cujo *schema* de dados é armazenado pelo SGBD *MySQL* 5.0.

O *jUDDI* e o *MySQL* serão os responsáveis por armazenar as marcações semânticas dos serviços e serão encarregados de retornar o conjunto de serviços que possam atender à requisição do usuário ao ser consultado pelo *matchmaker,* durante o processo de descoberta.

5.8.4 Descrição Detalhada dos Componentes

5.8.4.1 Cliente

Os clientes poderão acessar o sistema SOMM tanto via *browser* como através de uma aplicação feita em *JAVA ME*. Pois, como a arquitetura foi implementada com *Web Services* a camada cliente necessita apenas fazer uma chamada ao serviço sem precisar se preocupar com a camada de negócios, se preocupando apenas com a *interface* do sistema.

Para criar a versão cliente do sistema para testes, foi utilizada a ferramenta *IDE Netbeans* 6.0 (Netbeans, 2008). Essa ferramenta foi escolhida pela facilidade de manuseio e um grande suporte a construção de arquiteturas *Web* e aplicações *JAVA ME*. Ao compilar a aplicação cliente, a própria ferramenta faz a integração com o servidor *Web Apache Tomcat* 5.5 (Apache Tomcat, 2008). Esse servidor foi escolhido por ser o mesmo utilizado pela API usada no sistema.

5.8.4.2 WsPrincipal

Este serviço é responsável por fazer a abstração e integração dos outros serviços do sistema, além de conter funções de configurações do sistema e informações de ajuda ao cliente. Sua implementação foi feita usando a tecnologia de *JavaServer Pages Technology* (*JSP*) e *Servlets*. Assim o código feito na linguagem *Java* fica separado do código *HTML*. Portanto, Proporciona uma melhora o reuso do código e facilidade na construção de uma arquitetura *Model-View-Control* (*MVC*), que garante que a separação entre a *interface*, controle de fluxo e regra de negócio. Assim, cada tipo de componente executa uma determinada tarefa.

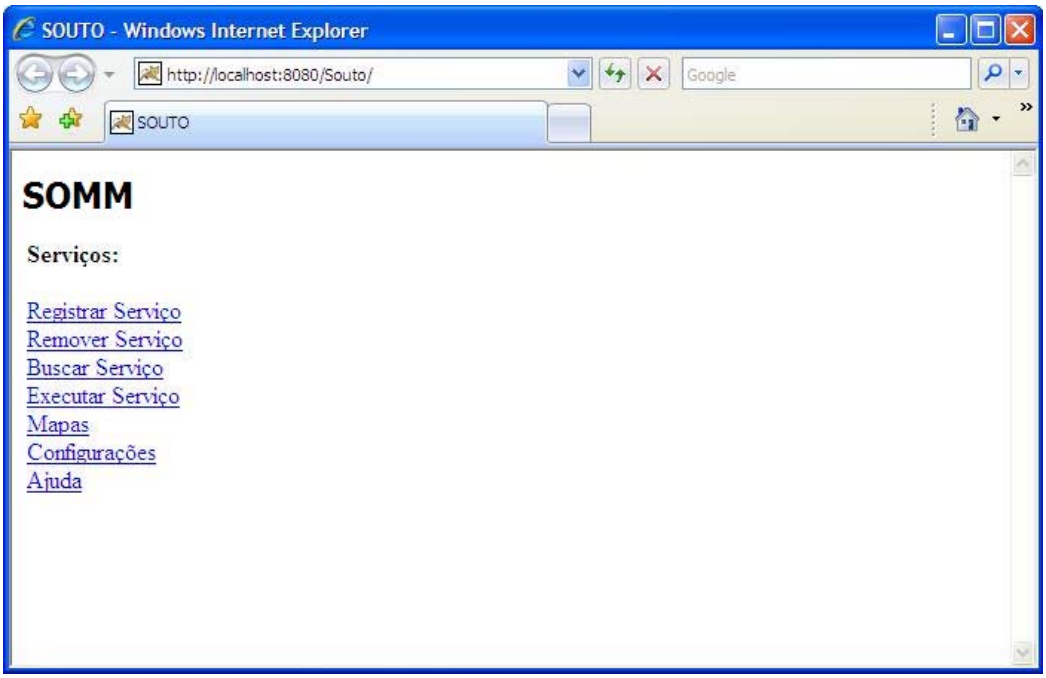


Figura 25 - Página Principal.

5.8.4.3 WsAnuncios

O WsAnuncios é responsável por registrar o serviço representado em uma ontologia em *OWL-S*, através do seu endereço do *Profile*. Assim, o *Web Service* busca os parâmetros de entrada e saída do serviço e o registra na base de dados implementada através do *OWL-S UDDI Matchmaker.*

Para usar os serviços da API, utilizou-se API OWL-S Matchmaker Client 1.1 (*OWL-S Matchmaker Client*, 2008). Esta API é o cliente para o *OWL-S UDDI Matchmaker (OWL-S UDDI Matchmaker, 2008)*, atualmente na versão 1.3 e disponível no mesmo site que o cliente, este é a implementação do *UDDI* semântico descrito pelo trabalho de (Srinivasan et al., 2004).

O *OWL-S UDDI Matchmaker* é implementado em *Java* e consiste de um repositório *UDDI*, cuja implementação é dada pelo *jUDDI*, versão 0.8.0 que possui o *schema* de dados armazenado pelo *MySQL* versão 5.0. Essa ferramenta utiliza o *Jena* (*Jena*, 2008), versão 2.1, como mecanismo de inferência para realizar o casamento entre as entradas e saídas da requisição do usuário com os conceitos das ontologias de domínio que representam as entradas e saídas do serviço semântico.

A escolha dessa API se dá pela grande quantidade de documentação existente, pela existência de uma lista de discussão ativa e participante, e ainda pela sua constante evolução. Quanto à escolha de *OWL-S*, esta se dá

pela sua ampla aceitação e evolução dentro do *W3C* (*W3C*, 2008) e pelo seu crescente uso como linguagem para construção de ontologias na *Web*.

O processo de registro do serviço consiste das seguintes fases: na primeira, o provedor de serviços deve anotar semanticamente seus serviços, através de uma ferramenta de criação de ontologias, como o *Protégé* (*Protégé*, 2008).

O *Protégé* é uma ferramenta desenvolvida por um grupo da universidade de *Stanford* para auxiliar na construção e edição de ontologias *OWL.* A ferramenta foi desenvolvida com a linguagem *Java,* sendo totalmente gratuita e livre (*open source*).

Em seguida, o provedor de serviços deverá entrar com a URL da ontologia *OWL-S Profile* referente ao serviço anotado. Ao recuperar o anúncio do serviço, o serviço de registro, através do *matchmaker,* irá realizar um processamento semântico em todo o arquivo *OWL-S Profile*, mapeando as informações passadas como as entradas, saídas, pré-condições e efeitos (IOPES), a categoria, classificação do serviço, bem como, informações sobre a organização que o desenvolveu. Por fim, estas informações são persistidas nos registros do *UDDI*. A **Figura 26** e a **Figura 27** exemplificam o processo do registro do serviço.

O processo de remoção do serviço na base de dados também é muito simples. O agente precisa apenas digitar a URL da ontologia *OWL-S Profile* que ele deseja remover, como é exemplificado nas **Figura 26**, **Figura 28** e na **Figura 29***.*

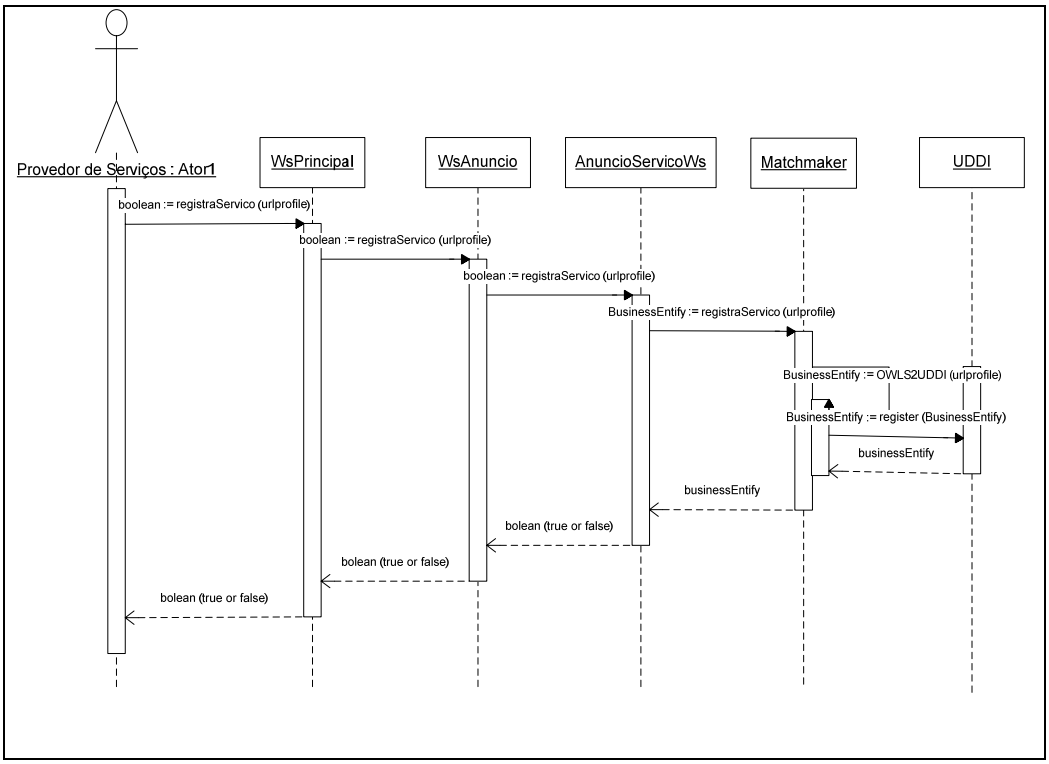


Figura 26 - Diagrama de seqüência: Registro de Serviço.

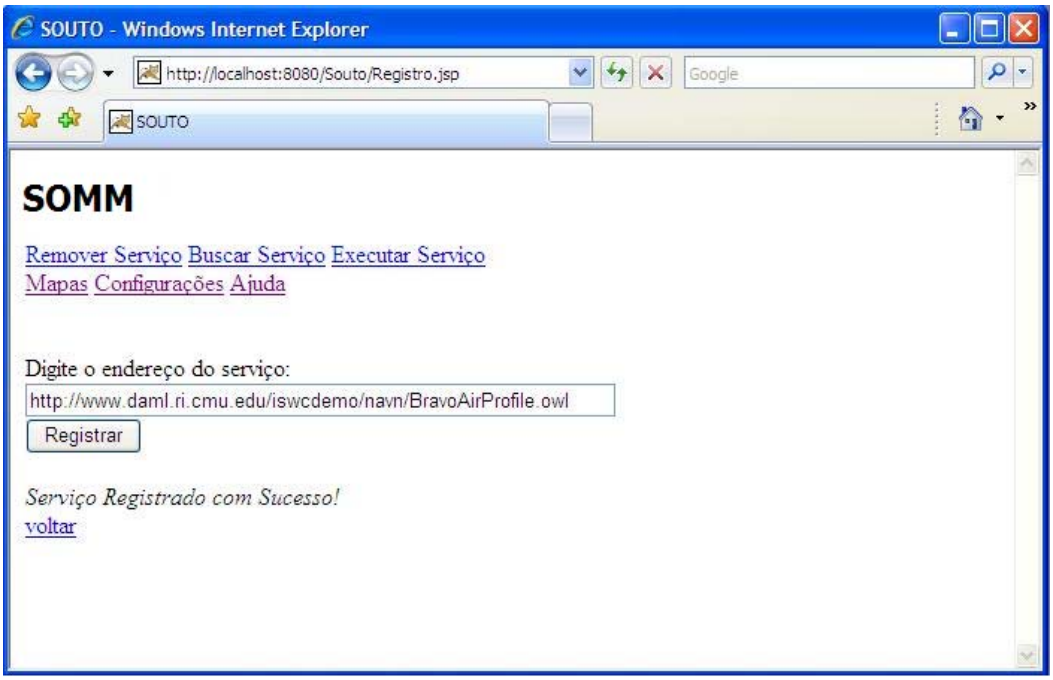


Figura 27 - Resultado do registro de serviços no serviço semântico.

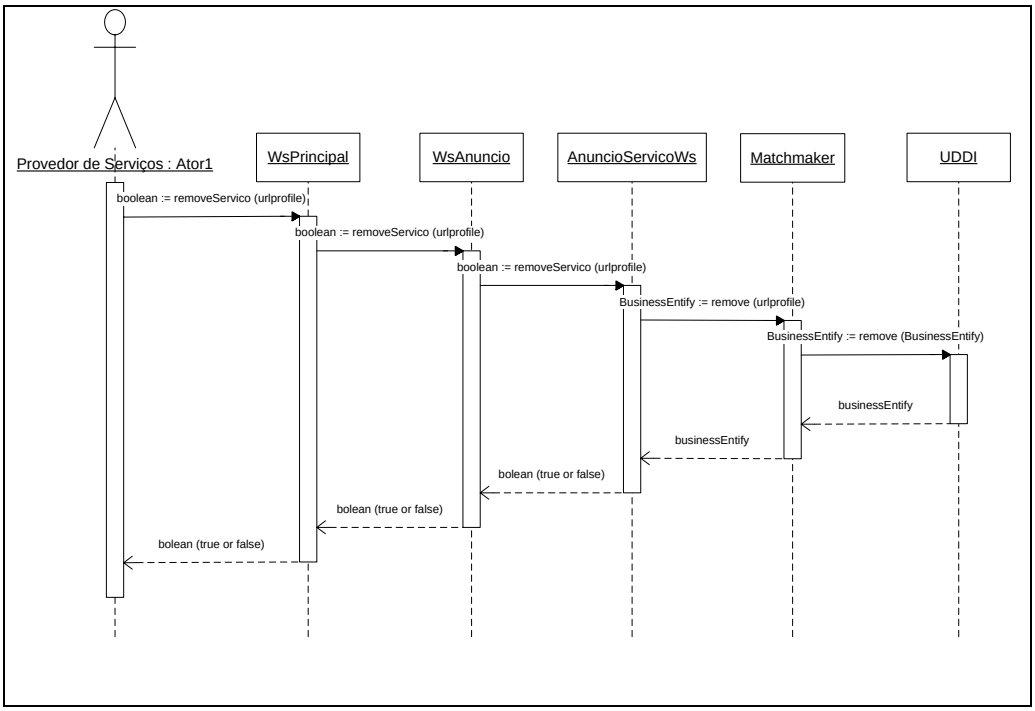


Figura 28 - Diagrama de seqüência: Remoção de um serviço.

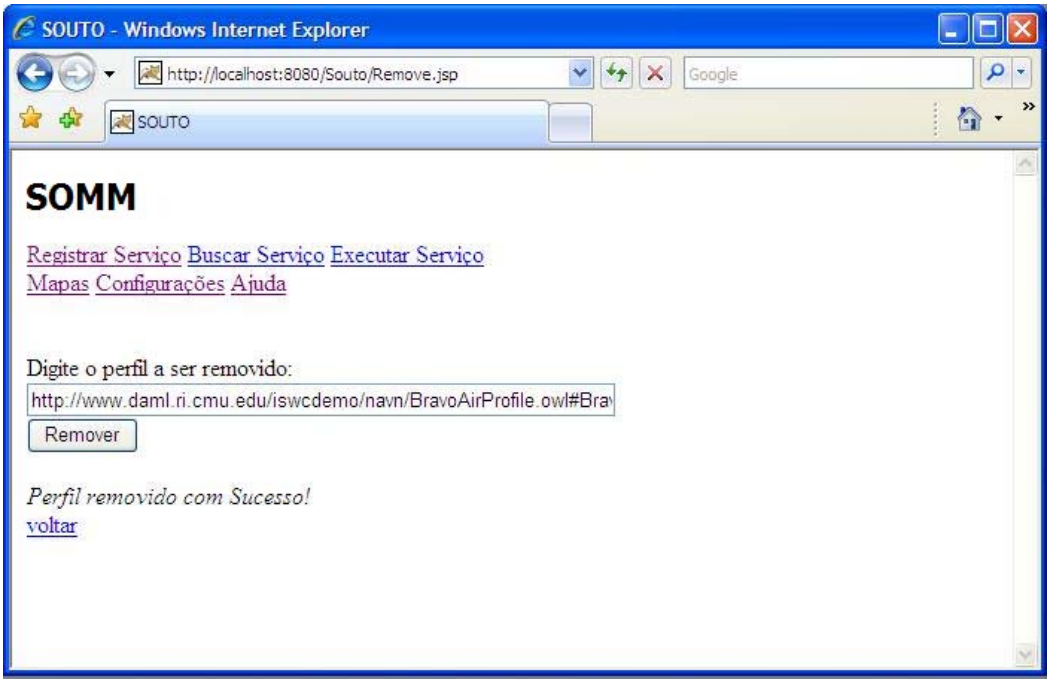


Figura 29 - Resultado da remoção de serviços no serviço semântico.

5.8.4.4 WsDescobertadeServiços

O WsDescobertadeServiços é responsável por descobrir de forma automática os serviços que possam atender à requisição do usuário. Esse serviço realiza apenas a descoberta de serviços simples, ou seja, que

possam atender diretamente à requisição do usuário sem a necessidade de composição de serviços devido a dificuldade de implementação e manipulação dos mesmos.

Para fazer a descoberta dos serviços é necessário que o usuário forneça os valores de entrada e saída do serviço que desejado do serviço de *matchmaker* do *UDDI* Semântico. Dessa forma são retornados todos os serviços que estão registrados na base de dados, conforme os dados que foram requisitados.

O *matchmaker* utiliza um algoritmo para realizar o casamento entre os conceitos passados na requisição e os conceitos que estão armazenados no *UDDI*, que trabalha com quatro níveis de filtros: exato, *plug-in, subsume* e falha explicados no capítulo 4.

Assim que o *matchmaker* finaliza a busca pelos serviços, este deverá retornar as informações para o serviço de descoberta, para que estas sejam processadas e retornadas à camada de apresentação, para serem visualizadas pelo usuário. A

Figura 30 e a **Figura 31** mostram o resultado da descoberta de serviços no serviço semântico. Nesse exemplo, os resultados são exibidos as *tags* em XML para que possa ser visto como é a definição do resultado, mas para o usuário final é realizado um tratamento desse resultado, para que se torne mais fácil sua compreensão.

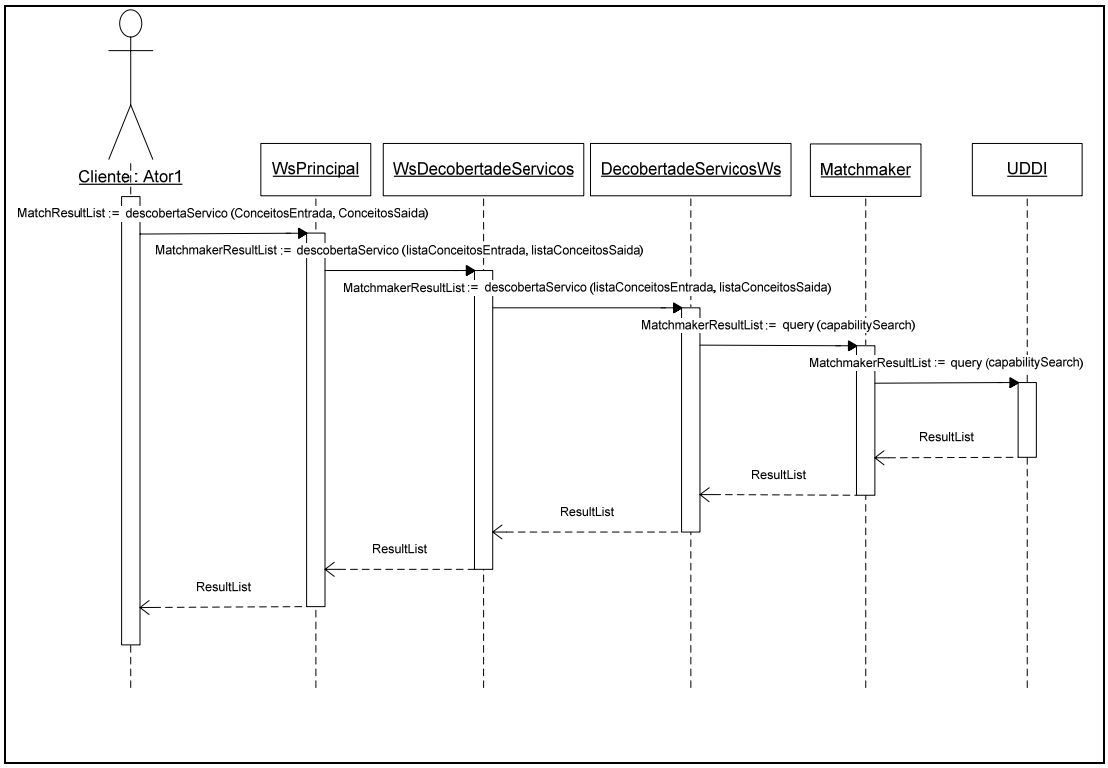


Figura 30 - Diagrama de seqüência: Descoberta de Serviços.

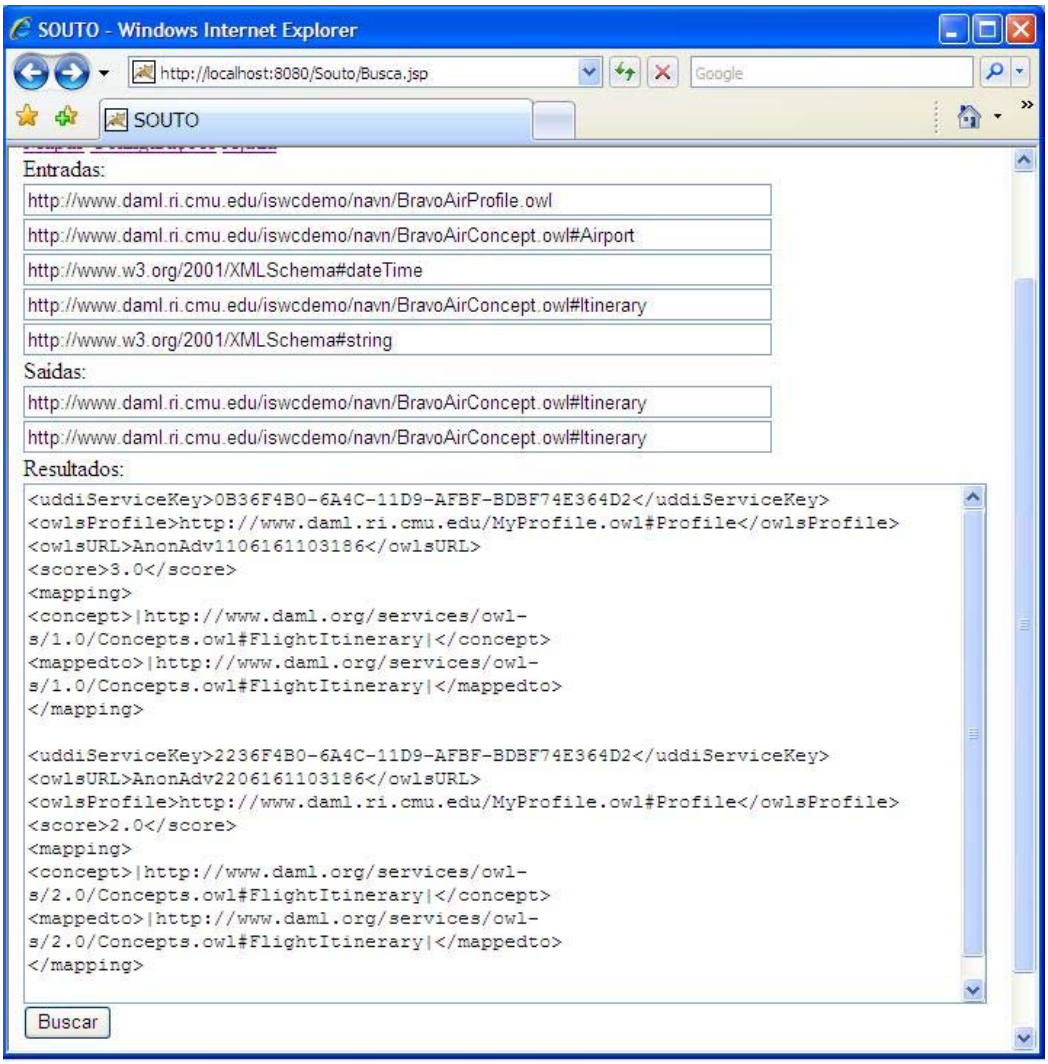


Figura 31 - Resultado da descoberta de serviços no SOMM.

5.8.4.5 WsExecucaoServicos

O WsExecucaoServicos é responsável por executar o serviço e retornar o resultado obtido para o usuário. Para desenvolver esse serviço foi o utilizado a OWL-S API 1.1.3 (*OWL-S* API, 2008) que é capaz de ler, executar e construir arquivos representados através de ontologias em formato *OWL-S*.

A *OWL-S* API possui, como principais características, a capacidade de invocar serviços atômicos que possuem a ontologia *OWL-S Grounding* (abstração para a descrição concreta do serviço) interligada com *WSDL* ou *UPnP* (descrições concretas de como acessar o serviço), assim como executar composições de serviços com estruturas de controle de execução como *Sequence, Unordered* e *Split*.

A execução de um serviço semântico acontece, apenas após acontecer à descoberta semântica do serviço. O resultado da

um conjunto de serviços semânticos que possuem o potencial de requisição do usuário. Os serviços a serem invocados devem ser selecionados pelo usuário, e este deve passar os valores dos entrada do serviço. Logo abaixo é apresentado o Diagrama de método de execução do serviço de Dicionário

Figura 32. A **Figura 33** mostra o resultado da execução do serviço de Dicionário. Na página do projeto que disponibiliza a OWL-S API possui alguns exemplos de serviços que foram utilizados para testar a arquitetura.

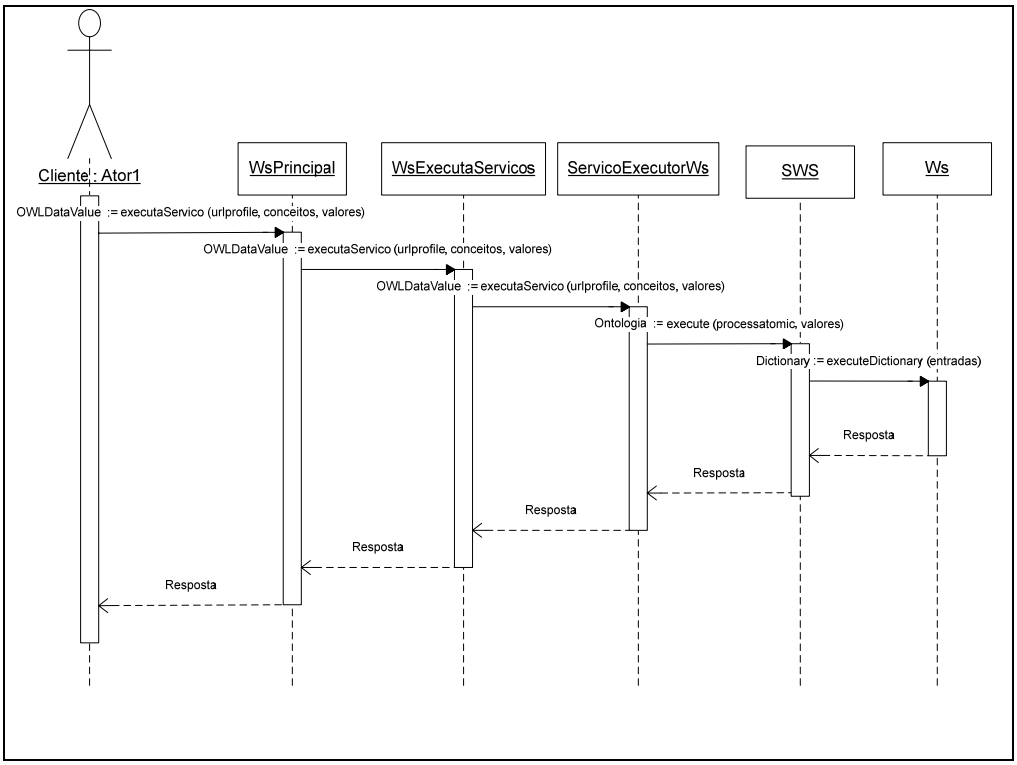


Figura 32 - Diagrama de Seqüência: Execução de serviços.

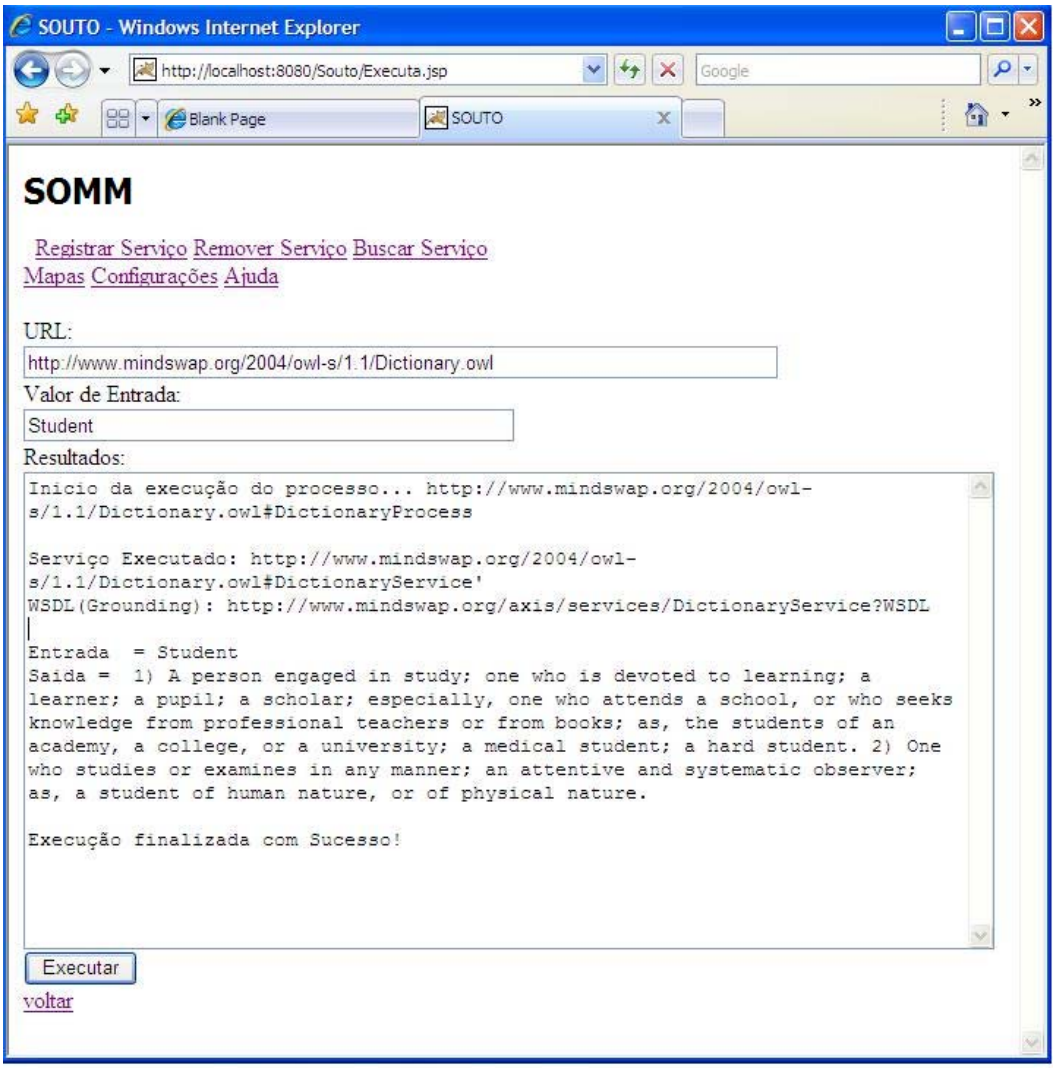


Figura 33 - Resultado da execução do serviço no serviço semântico.

5.8.4.6 WsPerfil

É o serviço que garante o registro do perfil do usuário que deve ser carregado assim que o usuário entrar no sistema. No perfil estarão as informações sobre as preferências e a localização do usuário que deverão ser preenchidas quando o usuário se cadastrar pela primeira vez no sistema.

5.8.4.7 WsLocalizacao

É o serviço que tem a tarefa de capturar a informação de localização do usuário e mostrar o mapa exibindo a localização dos serviços.

Para efetuar a exibição dos mapas, é necessária a implementação da *interface de* especificações *SIG* oferecidas pela *Open Geospatial Consortium* (*OGC*) (como *WMS*, *WFS* e *OpenLS*), mas que necessitam de algumas modificações, pois possuem sua própria *interface* de comunicação,

ou seja, não possuem uma *interface* padronizada de acordo com os *Web Services (OGC*, 2008*)*.

a) *OpenGIS Location Services (OpenLS*)

A especificação *OpenLS* (*OGC*, 2008) define um conjunto de *interface*s para o desenvolvimento de serviços baseados em localização, todos utilizando protocolos padrão *Web*. Os serviços especificados encontram-se descritos a seguir:

- **Serviço de Diretório**: provê acesso a um diretório online para localização de um determinado lugar, produto ou serviço.
- **Serviço de *Gateway***: identifica a posição geográfica de um determinado dispositivo móvel.
- **Serviço de Geocodificação/Geocodificação reversa**: identifica uma posição geográfica, dado o nome de um lugar ou endereço. Também funciona de forma reversa identificando um endereço completo dado uma posição geográfica.
- **Serviço de Apresentação de Mapas**: apresenta informações geográficas no terminal móvel. É usado para apresentar mapas destacando rotas entre dois pontos, pontos de interesse, áreas de interesse, localizações e/ou endereços.
- **Serviço de Determinação de Rotas**: determina a rota entre dois pontos informados pelo usuário. O usuário também pode, opcionalmente, informar pontos pelos quais a rota deve passar, rotas preferenciais (mais rápida, mais curta, menos tráfego, mais atrativa, etc.) e o modo de transporte.

b) *Web Map Service (WMS*)

A especificação *WMS* 1.1.1 (*OGC*, 2008) padroniza *interface*s que devem ser utilizadas por clientes para requisitar mapas aos servidores e também padroniza o modo como estes servidores devem descrever e retornar estes mapas.

Um servidor *Basic WMS* é capaz de:

- Gerar mapas georeferenciados (como uma imagem ou um conjunto de objetos gráficos).

- Responder perguntas sobre o conteúdo de um mapa, retornando informações sobre um determinado objeto (*feature*) do mapa.
- Descrever quais mapas ele pode produzir e quais podem ou não ser consultados, para que um cliente deste servidor saiba quais mapas podem ser requisitados.

Estes serviços podem ser requisitados pelo cliente utilizando as três *interface*s definidas pela especificação *WMS*:

- ***GetMap*** (obrigatória) para requisitar um mapa. Na requisição devem ser especificados parâmetros como o *layer*, a área que deve ser mapeada (*extent*), o sistema de coordenadas e nome do estilo.
- ***GetFeatureInfo*** (opcional) para consultar o mapa. Na requisição deve ser especificada a coordenada em que deve ser feita a consulta.
- ***GetCapabilities*** (obrigatória) para descrever os mapas.

Esta camada não foi considerada como prioridade neste projeto visto a existência de muitos projetos que já tratam esse problema de maneira satisfatória utilizando Sistemas de informação Geográfica (SIG) e assim a implementação foi proposta como trabalhos futuros.

5.9 Exemplo de Uso.

Esta seção descreve dois cenários de testes usados para avaliar as funcionalidades oferecidas pelos serviços da arquitetura SOMM. Esses cenários consistem de alguns usuários fictícios com relacionamentos entre si, alguns produtos e serviços cadastrados como veículos e dicionários. O primeiro apresenta um estudo de caso onde o usuário necessita efetuar a busca de um serviço semântico. O segundo apresenta uma aplicação onde o usuário necessita, além de efetuar a busca, efetuar a execução automática de um serviço semântico.

5.9.1 Sistema de busca de serviços semânticos.

Atualmente, existe uma enorme demanda por aplicações que possam ser acessadas por dispositivos móveis, pois os usuários estão utilizando cada vez mais dispositivos móveis para acessar a Web e através dela

efetuar pesquisas sobre produtos ou localização de serviços através de mapas.

O grande problema é que, apesar da grande evolução desses dispositivos ao longo dos últimos anos, ainda existem muitas limitações dos mesmos em função do seu tamanho e recursos disponíveis (processamento, memória, interfaces de entrada e saída etc.). Outro problema é em relação aos mecanismos de busca existentes que efetuam buscas sintáticas. Estes muitas vezes não conseguirão encontrar o veículo desejado, se realizarem a pesquisa pelo veículo utilizando o nome do modelo de forma diferente da cadastrada naquele sistema.

A arquitetura SOMM foi criada para tentar melhorar essa busca por serviços. Para alcançar esse objetivo, ela foi construída com base nos conceitos da Web Semântica onde as informações dos serviços são “anotados”, descrevendo os dados com base na semântica que possuem, **Figura 34.** Além disso, ela é capaz de armazenar previamente as informações sobre o perfil do usuário e capaz de executar serviços web.

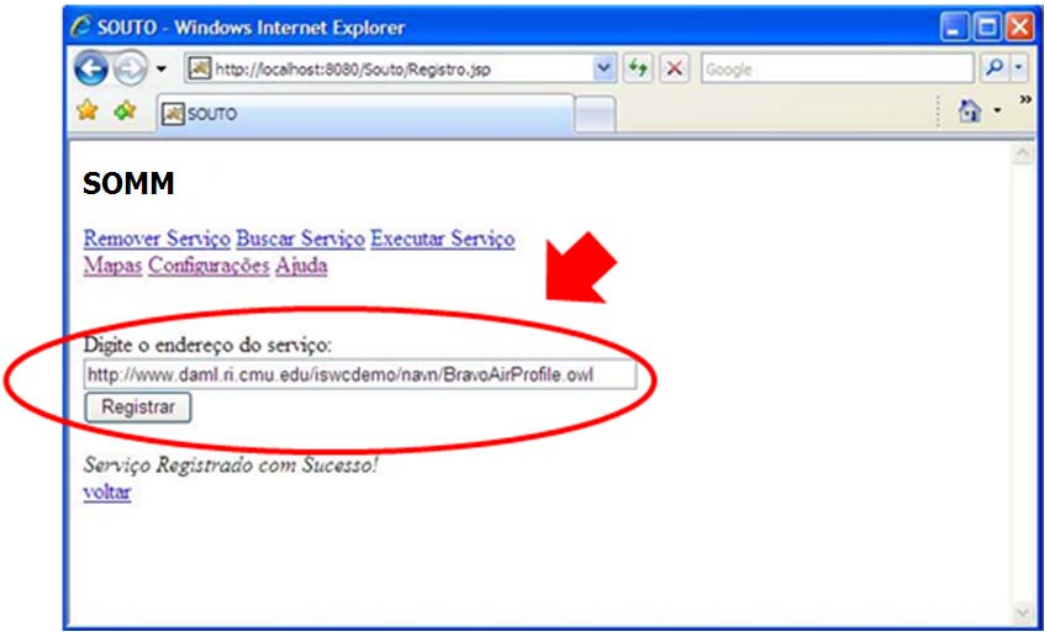


Figura 34 - Exemplo de uma anotação do serviço de Itinerários aéreos.

Ao utilizar a arquitetura SOMM, uma empresa pode fazer a anotação semântica do serviço, através de uma ontologia de serviços, como OWL-S. A ontologia de serviços oferece a automatização da descoberta e invocação de Serviços Web Semânticos para que um agente de software possa executar o serviço sem a necessidade de intervenção humana.

Dessa forma, quando um usuário realiza uma pesquisa por informações de um itinerário de um vôo de avião, ele espera que o itinerário pesquisado seja encontrado e retornado **Figura 35**. Porém, ao utilizar as ferramentas de busca padrão, como as disponibilizadas por vários sistemas existentes, como Google, ou mais especificamente, sistemas de informações das empresas aéreas, provavelmente ele terá muita dificuldade de encontrar as informações corretas e terá que inserir uma grande quantidade de informações através do dispositivo móvel para conseguir concluir sua pesquisa.

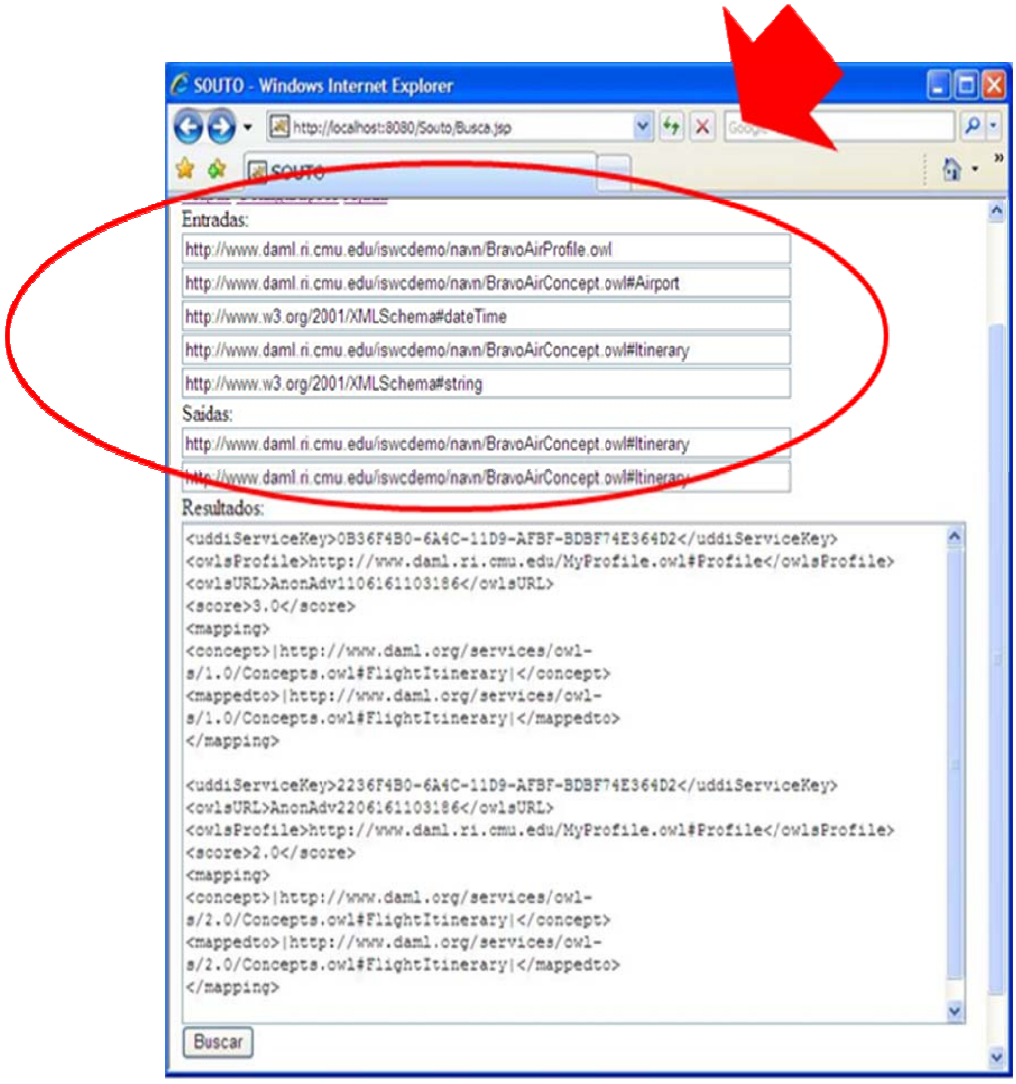


Figura 35 - Resultado de uma pesquisa sobre um itinerário aéreo.

Caso a pessoa utilize o SOMM, ela terá que cadastrar previamente o seu perfil que irá conter informações sobre suas preferências e interesses. Ao efetuar uma busca, além das informações que ele irá inserir como condições de entrada, o seu perfil também será considerado na busca

semântica. Portanto, a busca não será mais sintática diminuindo a quantidade de interações do usuário com o dispositivo para que ele necessite inserir uma grande quantidade de informações.

5.9.2 Sistema de busca e execução de serviços semânticos.

Em função da grande evolução das redes de comunicação, as pessoas estão acessando cada vez mais a internet através de dispositivos móveis, pois ela pode efetuar essa navegação onde ela estiver sem a necessidade de utilizar um computador *desktop*. Assim, além da pesquisa por informações sobre noticias e leitura de e-mails, as pessoas utilizam a busca por serviços e a execução de serviços diversos.

Como os mecanismos mais comuns utilizam a busca sintática para poder disponibilizar as informações para o usuário, muitas vezes o usuário terá dificuldade em encontrar o serviço desejado. Além disso, depois de encontrar o serviço desejado, ele terá que inserir uma grande quantidade de informações para poder executar esse serviço.

A vantagem de se utilizar o SOMM é que ele pode solucionar esses problemas, pois quando o provedor de serviços registra um novo serviço, ele faz a anotação semântica do serviço, através de uma ontologia de serviços, como OWL-S. A ontologia de serviços então oferece a automatização da descoberta e invocação de serviços web semânticos para que um agente de software possa executar o serviço sem a necessidade de intervenção humana. Assim, quando a pessoa faz uma busca semântica por um dicionário, por exemplo, **Figura 36**, após decidir qual será o serviço que irá querer, caso esse serviço seja atômico, ele irá ser executado automaticamente.

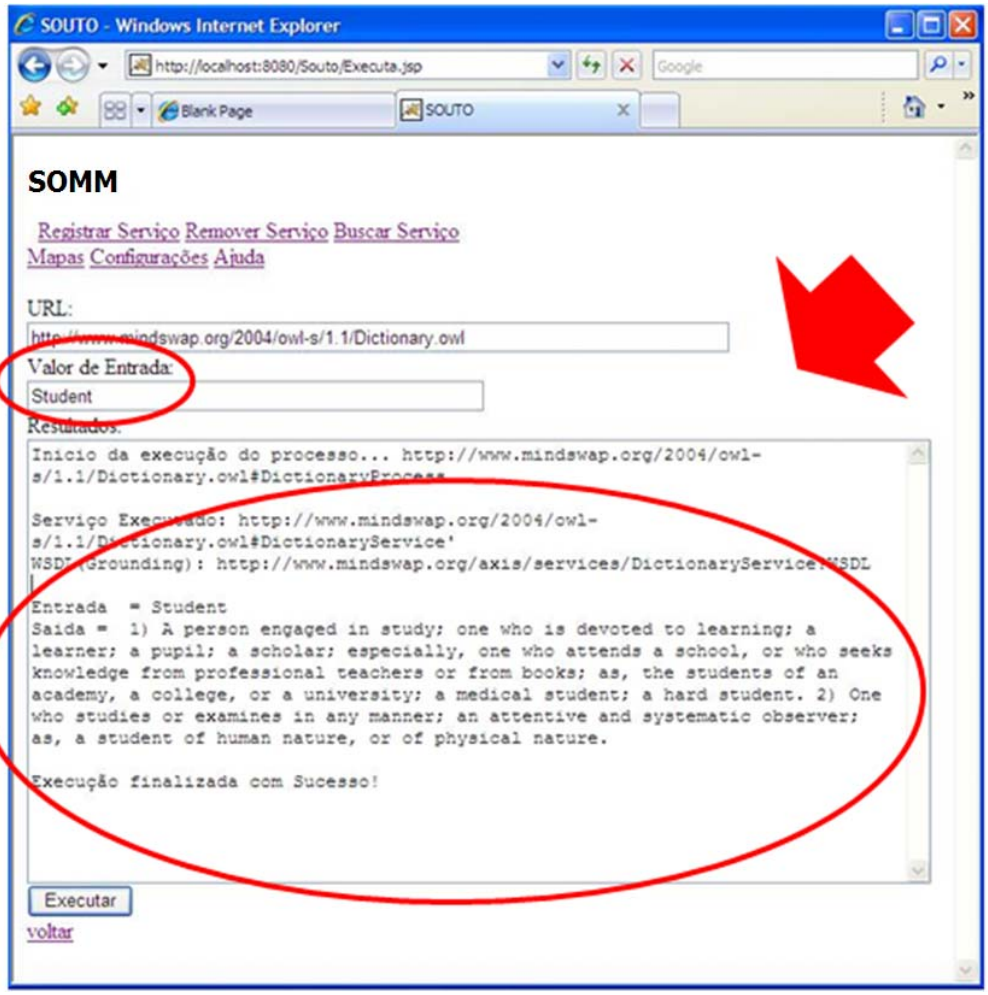


Figura 36 - Resultado de uma pesquisa em um dicionário sobre a palavra "Student".

5.10 Considerações Finais e Resultados

Neste capítulo, foram descritos aspectos da arquitetura de um sistema para aplicações cientes de contexto, o SOMM. Foi exibida a arquitetura geral do sistema, apresentando todos os componentes envolvidos. De acordo com o que foi desenvolvido, é possível realizar uma análise sobre os resultados da arquitetura SOMM.

O SOMM é um sistema capaz de facilitar o acesso a serviços semânticos para o usuário de dispositivos móveis, pois o mesmo não necessita saber o nome correto do serviço esperado, pois a busca é realizada semanticamente e de forma automática. Assim, o acesso poderá ser feito através dos valores de entrada fornecidos pelo usuário, de forma a automática apenas carregando os dados do perfil armazenado no sistema.

Outra característica muito relevante é que os serviços são executados também de forma automática, sem que o usuário necessite realizar muitas interações com sistema.

A utilização das *API's* foi de grande valor para o projeto devido a dificuldade de manipulação e execução de serviços semânticos devido ao fato das pesquisas nessa área ainda estarem imaturas e ao tempo disponível para realização do trabalho.

Os resultados mostraram que a arquitetura é bastante viável e que ela atende ao objetivo almejado. Novos requisitos poder ser implementados, tais como:

- O sistema ainda não é capaz de mapear e carregar o perfil do usuário de forma automática para a realização das buscas. Deve ser implementado uma função para verificar qual é o perfil do usuário logado e já forneça os serviços de acordo com o seu perfil;
- A *interface* gráfica é bastante complexa para que um usuário possa utilizá-la. Um estudo na área de *interface* homem-máquina seria interessante para descobrir a melhor forma para especificar essas regras de contexto. A *interface* do programa do dispositivo móvel também necessita ser aperfeiçoada.

6 CONCLUSÃO

6.1 Introdução

Construir um serviço que analise todas as informações de contexto é bastante complexo. Nesta dissertação, os principais contextos (dentre eles, o perfil do usuário) foram levados em conta para montar uma arquitetura baseada em dispositivos móveis que possa disponibilizar e facilitar o acesso a serviços personalizados para o usuário. Nas próximas seções, são discutidas as contribuições feitas, as dificuldades encontradas e as sugestões de trabalhos futuros.

6.2 Contribuições e Resultados

A principal contribuição desta dissertação é o desenvolvimento do SOMM, uma arquitetura baseada em *Serviços da Web* Semântica que proporciona várias funcionalidades na área de computação ciente de contexto. Dessa forma, conseguimos concluir o principal objetivo deste trabalho. Mas, por motivo de tempo não conseguimos atingir o terceiro e quarto objetivos específicos cidatos na seção 1.2.

O terceiro objetivo era crar um algoritmo de matching próprio do SOMM, mas como a API utilizada já possuía um algoritmo concolidado preferimos usá-la como padrão do SOMM também. O quarto objetivo não foi realizado em função da dificuldade de configuração do SOMM para executá-la em um novo servidor, devido as API's que ela utiliza.

A **Tabela 12** mostra uma comparação de características citadas na seção 4.6 entre o SOMM e as demais arquiteturas estudadas:

Características / Sistemas	1	2	3	4	5	6	7
SOCAM	X	X	X				
SOUPA	X	X					
COMPASS2008	X	Apenas localização e perfil	X	X	X	Apenas por tipo do produto	
Nexus	X	Apenas a localização		X			
FieldMap		Apenas a localização		X			
SOMM	X	Todos, menos o contexto computacional	X	X	X	X	X

Tabela 12 - Tabela de comparação das características das infra-estruturas estudadas, incluindo o SOMM.

Os serviços oferecidos por essa arquitetura são:

- Serviços web de anúncio de serviços que permitem que usuários possam buscar e executar serviços de forma que o anúncio ou a busca pelo serviço é realizado de forma dinâmica, enquanto ele se movimenta pela cidade. Para isto, faz-se necessário que o provedor de serviços anote semanticamente as informações dos serviços web, através do uso de ontologias de domínio e de serviços (como OWL e OWL-S, respectivamente).
- Um modelo de representação do contexto usando ontologias, baseado nas principais características do contexto do usuário e do ambiente.
- Um serviço de mapas, com os pontos de interesse de acordo com o perfil do usuário.

Como se trata de uma arquitetura orientada a serviço, esses serviços podem ser acessados independentemente. Dessa forma, usuários que só necessitam visualizar mapas, por exemplo, irão consultar apenas um serviço. Outros serviços podem ser adicionados à arquitetura, desde que sigam o padrão *Web Service* da *W3C*. Por exemplo, seria possível acrescentar um serviço que anotasse o histórico do perfil do usuário. Outra vantagem em desenvolver a arquitetura em *Web Services* é que ela permite

que outros tipos de clientes, além do browser, possam acessar os serviços de forma padronizada, sem que seja necessário acrescentar novos tipos de *interface*s para os vários tipos de clientes. Porém, devido às limitações dos dispositivos móveis, nem todas as funcionalidades oferecidas pelos serviços podem ser processadas por estes, como é o caso da criação de uma ontologia de domínio de um serviço.

Como fruto do nosso trabalho obtivemos três publicações (Mendes et al, 2008) (Mendes, 2008) (Mendes, 2007) em eventos importantes no Brasil e Exterior. Portanto, demonstra que o assunto abordado tem grande potencial para pesquisas futuras.

6.3 Limitações e Dificuldades Encontradas

Dentre as considerações a serem feitas sobre o atual estágio da *Web* Semântica e o desenvolvimento dos serviços web semânticas, observa-se que muita coisa ainda precisa ser feita, principalmente, pelo fato de ser uma área em crescente desenvolvimento científico e ainda não existir para o desenvolvedor, um ambiente de desenvolvimento integrado e suficientemente maduro. Muitas vezes foi necessário, descobrir e utilizar componentes de terceiros, os quais nem sempre cooperam bem entre si.

Outra dificuldade encontrada refere-se à bibliografia existente ainda bastante recente e a maior fonte de informações disponíveis muitas vezes não oferecem informações suficientes que facilitem a implementação dos sistemas.

De fato, o que constatamos é que poucos trabalhos são divulgados de forma completa e documentados. Ao buscar por ontologias para reuso em nossa aplicação, por exemplo, constatamos que existem poucas ontologias disponíveis para uso. Dentre essas, muitas são apropriadas como exemplos para aprendizado, mas não para uso efetivo.

Dessa forma, torna-se muito difícil fazer uma análise comparativa ou um ensaio baseado em vocabulários reais. Apesar das discussões em torno de ontologias de tarefa na literatura, não se encontram exemplos disponíveis para uso. Da mesma forma, a falta de documentação muitas vezes dificulta sobremaneira o entendimento de ontologias que conseguimos utilizar,

tornando difícil delinear o escopo de seus descritores em relação aos domínios que estes se propõem a descrever.

Por outro lado, é verdade que existe uma comunidade atuante e cooperativa, disposta a se ajudar e a compartilhar suas dúvidas e experiências. Foi o que constatamos ao interagir com as listas de discussão dos usuários *Jena* (*Jena*, 2008) e *Protégé* (*Protégé*, 2008).

Contudo, o desenvolvimento de projetos que incorporam as tecnologias da *Web* Semântica tem proporcionado a construção de sistemas inteligentes e a automatização de tarefas que, até então, não podiam ser realizadas pelos computadores sem o auxílio humano. Esta mudança gradativa, conforme vislumbrado por Tim Berners-Lee (Berners-Lee et al., 2001), tem tornado a Web, outrora de documentos espalhados, em uma web onde os conhecimentos encontram-se inter-relacionados.

6.4 Considerações Finais e Trabalhos Futuros

A área de Computação Ciente de Contexto é bastante vasta, possuindo margem para a implementação de uma série de trabalhos. A partir dos resultados obtidos com SOMM, podem-se identificar algumas questões importantes para o aperfeiçoamento e extensão deste trabalho. Seguem algumas sugestões de funcionalidades ainda incompletas ou não implementadas que podem ser realizadas em trabalhos futuros:

- A implementação de um cliente utilizando o *JAVA ME*, com o objetivo de criar uma *interface* para dispositivos móveis.
- Um ponto importante para a arquitetura é a questão da composição de serviços. No SOMM, a composição de serviços ajudaria a selecionar, dinamicamente, serviços que enriqueceriam a informação para o cliente. Por exemplo, se um usuário deseja visualizar um mapa com os pontos de ônibus próximos, juntamente com os seus horários, dificilmente o sistema irá encontrar um *Web Service* que ofereça os dois serviços. No entanto, um sistema de composição poderia procurar por um serviço de mapas e outro serviço de paradas de ônibus e apresentá-los ao usuário, de forma que este não saberia que no fundo se trata de dois *Web Services* diferentes.

- Utilização de um ambiente de desenvolvimento semântico integrado, que realize desde a criação de ontologias de domínio e serviços até a execução de serviços semânticos. Dentre os possíveis candidatos, destaca-se o *Protégé* que é um dos *softwar*es mais utilizados pela comunidade científica da *Web* Semântica.
- Manutenção automática de contexto e o perfil do usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

Alonso, G., Casati, F., Kuno, H., Machiraju, V., *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, Berlin Heidelberg, New York, 2004.

Andersen, K. V. B., Cheng, M., Klitgaard-Nielsen, R., *Online Aalborg Guide – Development of a Location-Based Service*. Technical report, Aalborg Universitet, 2003.

Apache Tomcat, *Apache Tomcat*. Disponível <http://tomcat.apache.org/>. Acessado em: abril de 2008.

Arbter, B., Bitzer, F., Ressel, W., *Modeling and Simulation of Mobility in Ubiquitous Computing*. In: Möhlenbrink, W. (ed.), Englmann, F.C., Friedrich, M., Martin, U., Hangleiter, U.: FOVUS - Networks for Mobility, Stuttgart, Alemanha, 2004.

Basiura, Russ et al., Professional ASP.NET *Web* services. Tradução de Mário Moro Fecchio. São Paulo: Pearson Education, 2003.

Berners-Lee, T., Hendler, J., Lassila, O., *The semantic web – a new form of the Web content that is meaningful to computer will unleash a revolvution of new possibilities*. Scientific American, May 17, 2001.

Bharat, R., and Minakakis, L., *Evolution of Mobile Location based Services*. Communications of the ACM, v.46 n.12, New York, 2003.

Booth, David., *Web* Service architecture. *W3C Working Draft*. Disponível em <http://www.w3.org/TR/2003/WD-ws-arch-20030808>. Acessado em: dezembro de 2007.

Brickley, D., Miller, L., *FOAF vocabulary specification*. Disponível em: <http://xmlns.com/foaf/0.1/>. Acessado em: 15 janeiro de 2007.

Broens, T., Pokraev, S., van Sinderen, M., Koolwaaij, J., and Costa, P. D., *Context-aware, Ontology-Based Service Discovery*. In Proceedings of the European Symposium on Ambient Intelligence (EUSAI'04) (EindHoven, The Netherlands, November, 2004). Springer, 2004.

Brown, P.J., Bovey, J.D., Chen, X., *Context-aware Applications: From the Laboratory to the Marketplace*. IEEE Personal Communications, Volume 4, Issue 5, páginas 58-64, 1997.

Chen, G., Kortz, D., *A Survey of Context-aware Mobile Computing Research. Technical Report TR2000-381*, Dartmouth College, November, 2000.

Chen, H. et al., *SOUPA: Standard Ontology for Ubiquitous and Pervasive Application*. In: INTERNATIONAL CONFERENCE ON MOBILE AND Ubiquitous Systems: Networking and Services, Boston, 22-25 A, pp. 258-267, 2004.

Chen, H., Finin, T., Joshi, A., *An ontology for context aware pervasive computing environments*. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review, 2003.

Chen, H., Finin, T., Joshi, A., *An ontology for context aware pervasive communication tool using location and schedule information*, In IEEE Pervasive Computing, (Jan.-Mar. 2004), 2004.

Coyle, Frank P., *XML, Web services and the Data Revolution*. Reading Addison - Wesley Information Technology Series, 2002.

Davies, J., Fensel, D., Harmelen, F. van. *Towards the Semantic Web*. John Wiley & Sons, 2003.

Dextra, *Boletim Dextra 02. Web Services na integração de sistemas corporativos.* Disponível em <http://www.dextra.com.br/empresa/boletim/0302-02/02tecnologia.htm>. Acessado em: abril de 2008.

Dey, A. K., Abowd, G. D., *Towards a Better Understanding of Context and ContextAwareness*. In: CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness, Hague, Netherlands, April, 2000.

Dürr, F., Höhnle, N., Nicklas, D., Becker, C., Rothermel, K., *Nexus - A Platform for Context-aware Applications*. In Proceedings GI-Fachgespräch “Ortsbezogene Dienste“, Hagen, Alemanha, 2004.

Lenat, D. B., Guha, R. V., *Building Large Knowledge-Based Systems*: *Representation and Inference in the Cyc Project*. Addison-Wesley, 1990.

Endrei, Mark et al., *Patterns: service-oriented architecture and web services*. [s.l.]: IBM, 364 p., 2004.

Feng, L., Apers, P.M.G., Jonker, W., *Towards Context-aware Data Management for Ambient Intelligence.* In: Proc. of the 15th Intl. Conf. on Database and Expert Systems Applications, Zaragoza, Espanha, 2004.

Ferris, Christopher, Farrell, Joel, *What Are Web services?.* In Communications of the ACM, Vol.46 n. 6, Junho, 2003.

FieldMap, *FieldMap 3.3.6 Manual*. Disponível em: <http://www.piranesi.dyndns.org/FieldMap/manual/>. Acessado em: dezembro de 2007.

Gu, T., Pung, H. K., Zhang, D. Q., *A Middleware for Building Context-aware Mobile Services.* In Proceedings of IEEE Vehicular Technology Conference, Milan, Italy, 2004.

Hansen, Roseli., *Glue script: Uma linguagem específica de domínio para composição de web services*. Dissertação de Mestrado, Universidade do Vale do Rio dos Sinos - Unisinos. São Leopoldo, 2003.

Held, A., Buchholz, S., Schill, A., *Modeling of Context Information for Pervasive Computing Applications*. In: World Multiconference On Systemics, Orlando, 2002.

Jena, *A Semantic Web framework for Java*. Disponível <http://jena.sourceforge.net/>. Acessado em: abril de 2008.

Kagal, L., Finin, T., Joshi, A., *A Policy Based Approach to Security for the Semantic Web*. In: International Semantic *Web* Conference (ISWC2003), 2, Florida, 2003.

Kaul, A., *On Line Matching of Semantic Web Services in Ambient Intelligence Environments*, Master's Thesis / Diplomarbeit, Mat. No. 262364, March 23rd, Germany, 2006.

Martin, D. et al., *OWL-S: semantic markup for web services 1.0*. Disponível em: <http://www.daml.org/services/owl-s/1.1/>. Acessado em: 12 abr. 2007.

Mendes, Thiago Souto, Rocha, M. N. Silva, Nidyana R. M. O. *A User Profile and Location-Based Services Matching Model Architecture*, Proc. of the IADIS International Conference Information Systems 2008, Carvoeiro, Algarve, Portugal, 2008.

Mendes, Thiago. Souto. *Um Modelo de Arquitetura de Matching entre Perfis de Usuários e Serviços Baseados em Localização*. In: <http://www.uniriotec.br/~sbsi2008>, 2008. IV Simpósio Brasileiro de Sistemas de Informação, Rio de Janeiro, 2008.

Mendes, Thiago Souto. *Um modelo de matching de perfis de usuários e serviços baseados em localização*. In: VII SIMPóS Mostra Cientifica da Pós Graduação. Viçosa, 2007.

Mostéfaoui, G. K., Pasquier-Rocha, J. and Brezillon, P., *Context-aware Computing: A Guide for the Pervasive Computing Community*. In Proceedings of the IEEE/ACS International Conference on Pervasive Services (ICPS´04). IEEE Computer Society, pp. 39-48, Washington, DC, USA. 2004.

Nakanishi, Y., Takahashi, K., Tsuji, T., Hakozoki, K., *ICAMS: A mobile communication tool using location and schedule information.* In IEEE Pervasive Computing, 3, 1, (Jan.-Mar. 2004), University of Electro-Communications, 2004.

Naumenko A. et al., *Using UDDI for Publishing Metadata of the Semantic Web*. In: M. Bramer and V. Terziyan (Eds.): Industrial Applications of Semantic *Web*, Proceedings of the 1st International IFIP/WG12.5 Working Conference IASW- 2005, Jyväskylä, Finland, Springer, IFIP, pp. 141-159, 2005.

Netbeans, *Netbeans IDE*. Disponível <http:// www.netbeans.org>. Acessado em: abril de 2008.

Newcomer, E., *Understanding web services: XML, WSDL, SOAP and UDDI*. Boston: Addison-Wesley, 2002.

Oellermann, Willian L., *Architecting Web services*. Apress, Berkely, CA, USA 2001.

OGC, *Open Geospatial Consortium*, Disponível em <http://www.opengis.org>. Acessado em: 29 de abril, 2008.

OWL-S API, Disponível em: < http://www.mindswap.org/2004/OWL-s/api/>, 2008.

OWL-S, *Matchmaker Client*, Disponível em: < http://projects.semwebcentral.org/projects/mm-client>. Acessado em: 10 de junho de 2008.

OWL-S, *UDDI Matchmaker*, Disponível em: < http://projects.semwebcentral.org/projects/OWL-S-UDDI-mm/>. Acessado em: 10 de junho de 2008.

Pan, F., Hobbs, J. R., *Time in OWL-S*. In: PROCEEDINGS OF AAAI-04 SPRING SYMPOSIUM ON SEMANTICWEB SERVICES, Stanford University, California, 2004.

Paolucci, M. et al., *Semantic Matching of Web Services Capabilities*. In: INTERNATIONAL SEMANTIC WEB CONFERENCE ON THE SEMANTIC WEB (ICWC), Londres: Springer-Verlag, 2002.

Pashtan, A., *Mobile Web Services*, Cambridge University Press, New York, NY, 2005

Perich. F., *MoGATU BDI Ontology*. University of Maryland, 2004. Disponível em: <http://mogatu.umbc.edu/bdi/20040131/>. Acessado em: 15 dez. 2007.

Pressman R., *Engenharia de Software*, 6a edição, McGraw-Hill Interamericana do Brasil, 2006.

Protégé, Disponível em: <http://protege.stanford.edu/>. Acessado em: 23 de abril de 2008.

Randell, D., Cui, Z., and Cohn, A., *A Spatial Logic Based on Regions and Connection.* in Principles of KnOWLedge Representation and Reasoning, Proceedings of the Third International Conference, B. Nebel, C. Rich and W. Swartout eds., San Mateo (CA): Morgan Kaufmann, pp. 165-176., 2002.

Schilit, B., Adams, N., Want, R., *Context-aware Computing Applications*. 1st International Workshop on Mobile Computing Systems and Applications, páginas 85-90, 1994.

Schilit, B., Theimer, M., *Disseminating Active Map Information to Mobile Hosts*. IEEE Network, páginas 22-32, New York, 1994.

Schiller, J., Voisard, A., *Location-Based Services*. Morgan Kaufmann, São Francisco, 2004.

Sinner, A, Kleemann T, Hessling, A von., *Semantic User Profiles and their Applications in a Mobile Environment*. Artificial Intelligence in Mobile Systems Journal, 2004.

Sommerville, Ian, *Engenharia de Software*, 8a edição, Pearson Addison-Wesley, 2007.

Souza, Vinícius., *SWservice: uma biblioteca para a escrita da língua brasileira de sinais baseada em web services*. Dissertação de Mestrado, Programa de Pós Graduação em Computação Aplicada - Unisinos, São Leopoldo, 2004.

Spiekermann, S., *General Aspects of Location-Based Services*. In Location-Based Services. Schiller, J., and Voisard, A. (eds). Morgan Kaufman, 2004.

Srinivasan, N., Paolucci, M., Sycara, K., *Adding OWL-S to UDDI, implementation and throughput*. In: INTERNATIONAL WORKSHOP ON the Semantic *Web* (ISWC2003), Ubiquitous Systems: Networking and Services, Boston, 2004.

UDDI.org. *White Papers*. Disponível em: <http://www.uddi.org/whitepapers.html>. Acessado em: outubro de 2007.

W3C, *World Wide Web Consortium*. *Web* Ontology Language. Disponível em: <http://www.w3.org/2004/owl/>. Acessado em: junho de 2008.

Weibenberg, N., Voisard, A., Gartmann, R., *Using Ontologies in Personalized Mobile Applications*, In Proceedings Intl. ACM GIS Conference, ACM Press, 2004.

Weiser, M., *Ubiquitous Computing*. Disponível em: <http://www.ubiq.com/weiser/>. Acessado em: junho de 2008.

Yu, S., Spaccapietra, S., Cullot, N., and Aufaure, M.-A. *User Profiles in Location based Services: Make Humans More Nomadic and Personalized*. In Proceedings of the IASTED International Conference on Databases and Applications (DBA'04) (Innsbruck, Austria, February, 2004). ACTA Press, 2004.