UNIVERSIDADE FEDERAL DA BAHIA

INSTITUTO DE MATEMÁTICA

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

Vanessa Aline dos Santos Sena

Incorporação da similaridade semântica no OWL-S Composer

Salvador

2009.2

Vanessa Aline dos Santos Sena

Incorporação da similaridade semântica no OWL-S Composer

Monografia apresentada ao Curso de graduação em Ciência da Computação, Departamento de Ciência da Computação, Instituto de Matemática, Universidade Federal da Bahia, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientadora: Profª . Daniela Barreiro Claro

Salvador

2009.2

RESUMO

Os serviços Web estão sendo cada vez mais utilizados em sistemas distribuídos. Isto se deve principalmente por causa das vantagens que esta tecnologia oferece como interoperabilidade, integração e baixo acoplamento. Contudo, um serviço sozinho, muitas vezes não atende as necessidades do usuário, sendo necessário criar composições. Além disso, a tarefa de encontrar um serviço pode se tornar uma atividade complicada para o usuário, se tiver que ser feita manualmente. Diversos trabalhos têm sido realizados nessa área com o intuito de facilitar o desenvolvimento e manipulação dos serviços. O *plugin OWL-S Composer*, para a plataforma Eclipse, permite que o usuário desenvolva composições de forma visual através de diagramas. Neste contexto, o presente trabalho propôs a incorporação da similaridade semântica para as composições de serviços desenvolvidas através da ferramenta *OWL-S Composer*. Esta similaridade semântica é baseada na etapa Semântico Funcional do algoritmo *OWL-S Discovery*. Desta forma, é possível ao usuário gerar serviços compostos através de diagramas e encontrar outros serviços similares.

Palavras-chave: Serviços Web Semânticos, OWL-S, Plugin, Eclipse, Descoberta Semântica.

ABSTRACT

Web services have been used a lot in distributed systems. This is because this technology has brought advantages like interoperability, integration and low coupling. However, sometimes only one service is not enough to meet client's requirements, they need to create compositions. There are some work been developed in this area in order to facilitate the development and manipulation of Web service. The OWL-S Composer plugin, for Eclipse, allow the user to develop compositions visually using diagrams. Besides that, the activity of finding a service can be complicated for the user, if he needs to do manually. In this context, this work proposed the incorporation of semantic similarity for the service compositions developed through the OWL-S Composer plugin. This semantic similarity was based in the functional semantic step of the OWL-S Discovery algorithm. This way, it is possible for the user to create composite services using diagram and find other similar services.

Keywords: Semantic Web Services, OWL-S, Plugin, Eclipse, Semantic Discovery.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CODE	CMU's OWL-S Development Environment,	p. 25
DoM	Degree of Match,	p. 21
EMF	Eclipse Modeling Framework Project,	p. 28
GEF	Graphical Editing Framework,	p. 28
GMF	Graphical Modeling Framework,	p. 28
JET	Java Emitter Templates,	p. 28
OWL	Web Ontology Language,	p. 15
OWL-S	Ontology Web Language for Service,	p. 15
SAWSDL	Semantic Annotations for WSDL,	p. 15
SOA	Service Oriented Architecture,	p. 13
SOAP	Simple Object Access Protocol,	p. 14
UDDI	Universal Description, Discovery, and Integration,	p. 14
URI	Uniform Resource Identifier,	p. 13
USDL	Universal Service-Semantics Description Language,	p. 26
WSDL	Web Services Description Language,	p. 14
WSMO	Web Service Modeling Ontology,	p. 15
WTP	Web Tool Plugin,	p. 27
XML	Extensible Markup Language,	p. 13

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A utilização de serviços Web está se tornando cada vez mais frequente em sistemas distribuídos. O serviço web é uma saída simples e prática para o problema da comunicação entre diferentes sistemas, permitindo que soluções já existentes possam ser reutilizadas através da web.

O aumento do uso de serviços Web, se deve ao fato desta tecnologia apresentar vantagens com relação a outras já existentes como CORBA (OMG, 1995) e DCOM (MICROSOFT, 1995). Uma grande vantagem é permitir a comunicação entre aplicações totalmente distintas, com a garantia de baixo acoplamento entre elas. Outra vantagem é que os serviços Web utilizam mensagens em formato XML, tornando-se independente de plataforma.

Apesar de existirem muitos serviços disponíveis na web, cada um, sozinho, tem uma utilidade limitada. Por esta razão, a noção de composição de serviços já está sendo usada para atingir um objetivo particular (CLARO; ALBERS; HAO, 2005). Uma composição deve ser descrita, indicando os serviços que fazem parte do processo e o fluxo de execução. Para descrever estas composições, podem ser utilizadas linguagens como OWL-S (MARTIN et al., 2004), WSMO (LAUSEN; POLLERES; ROMAN, 2005) e o SAWSDL (KOPECKY et al., 2007).

A web semântica, junto aos serviços Web, permitiu a automatização no processo de descoberta, invocação, composição e monitoramento dos serviços. Sem a web semântica, estes processos são manuais e consequentemente passíveis a um grande número de erros. Os serviços Web semânticos precisam ser descritos em linguagens específicas. A linguagem utilizada neste trabalho é a OWL-S, que é uma ontologia para representar serviços Web semânticos, baseada na OWL (BECHHOFER et al., 2004).

Diversos trabalhos têm sido propostos, com o intuito de facilitar a manipulação de composição de serviços Web semânticos. Como exemplo, o *CMU's OWL-s Development Environment* (CODE) (CMU, 2005), um *plugin* para o Eclipse, suporta todo o processo de desenvolvimento de

um serviço Web semântico, porém não é possível ter uma visualização completa da composição desses serviços.

Outro trabalho é o *plugin OWL-S Composer 1.0* (FONSECA; CLARO; LOPES, 2009), cuja funcionalidade principal é prover a criação de composição de serviços Web semânticos de forma visual, e a geração automática do código OWL-S desta composição. Desta forma, o *OWL-S Composer 1.0* torna fácil e intuitiva a atividade de criar e editar um documento OWL-S para composição de serviços Web. Esta ferramenta tem como vantagem, estar aliado a um ambiente integrado de desenvolvimento, que já oferece apoio na criação de serviços Web através do *plugin Web Tools Project* (WTP) (WTP, 2009).

O ciclo de vida de um serviço Web inclui a criação do serviço e a publicação em um repositório, atividades realizadas pelo provedor do serviço. O cliente realiza a localização do serviço que atenda à suas necessidades, e em seguida a invocação deste serviço direto no provedor. Das atividades citadas, o *OWL-S Composer 1.0* oferece suporte a criação do serviço Web através de diagramas.

1.2 PROPOSTA

Uma etapa importante no ciclo de vida de um serviço Web é a descoberta. O usuário precisa encontrar o serviço, antes de utilizá-lo, porém esta pode ser uma tarefa difícil visto o grande número de serviços disponíveis. A facilidade que a Web semântica ofereceu aos serviços Web possibilitou que a descoberta fosse feita de forma eficiente e automática. Um algoritmo de descoberta semântica foi proposto por (AMORIM, 2009), através da ferramenta *OWL-S Discovery*. Este algoritmo realiza descoberta em duas etapas, a Semântica Funcional e a Semântica Descritiva.

Diante deste cenário, este trabalho propõe uma nova versão do *plugin OWL-S Composer*. Esta versão permite ao usuário realizar uma busca semântica de forma que seja feita uma descoberta dos serviços disponíveis e sejam apresentadas algumas propostas de composições, semanticamente similares. Para realizar a descoberta, o *plugin* utiliza o algoritmo Semântico Funcional do *OWL-S Discovery*.

1.3 ESTRUTURA DO TRABALHO

Este trabalho encontra-se estruturado da seguinte forma: o capítulo 2 descreve a base teórica necessária para entender os conceitos de serviços Web semânticos, arquitetura orientada

a serviço, composição de serviços Web, além da linguagem de descrição de serviços Web semânticos utilizada neste trabalho; no capítulo 3 são apresentados os conceitos relacionados a descoberta de serviços Web semânticos e os algoritmos de descoberta semântica; no capítulo 4 são apresentados os trabalhos relacionados a este projeto, incluindo o *plugin OWL-S Composer*; o capítulo 5 é destinando ao desenvolvimento do trabalho, onde são explicadas todas as atividades realizadas, a nova arquitetura do *plugin OWL-S Composer* e um estudo de caso; no capitulo 6 é apresentada a validação do *OWL-S Composer 2.0*; por último, no capítulo 7 são apresentadas as conclusões finais, dificuldades encontradas e a indicação do que poderá ser desenvolvido como trabalhos futuros.

2 SERVIÇOS WEB

A Internet começou suportando interações com dados textuais e gráficos. As pessoas utilizam a Internet diariamente para ver cotações, fazer compras, e ler notícias. Este nível de interação atende a muitas das necessidades atuais. Mas a Web baseada em texto não suporta comunicação entre softwares da melhor maneira, principalmente transferência de grande volume de dados. Os serviços Web surgiram para desenvolver esta funcionalidade.

Os serviços Web estão mudando a maneira como as aplicações interagem na Web. Eles conectam programas que estão em pontos distantes do mundo, de forma mais eficiente e com menor custo. O resultado é uma melhor e mais produtiva comunicação entre empresas e clientes (NEWCOMER, 2002).

Um serviço Web é uma aplicação que pode ser acessada remotamente usando diferentes linguagens baseadas em XML. Normalmente, um serviço Web é identificado por uma URI, assim como qualquer outro site Web (POTTS; KOPACK, 2003). O uso do XML é uma das grandes vantagens dos serviços Web, pois os tornam independentes de linguagem, plataforma ou sistema operacional.

A arquitetura SOA (*Service Oriented Architecture*) propõe uma estrutura para a criação e utilização dos serviços Web, representada pela figura 1. Primeiramente é preciso que alguém ofereça o serviço, este é o papel do provedor de serviço. Então ele publica este serviço em um repositório, tornando-o disponível para que possa ser localizado. Um cliente realiza uma busca por um determinado serviço. Ao encontrar este serviço, o cliente tem acesso às informações de como utilizar o serviço. Essas informações incluem o endereço do provedor, através do qual o cliente pode fazer uma chamada direta ao provedor do serviço.

2.1 TECNOLOGIAS BÁSICAS

A arquitetura SOA utiliza tecnologias essenciais ao seu funcionamento. Padrões de representação das informações foram definidos para permitir a comunicação entre o cliente, reposi-

Figura 1: Arquitetura SOA (CLARO; MACêDO, 2008)

tório e provedor do serviço Web. Estes padrões serão apresentados a seguir:

- SOAP (*Simple Object Access Protocol*) é o protocolo de comunicação usado para troca de mensagens entre o cliente e o provedor do serviço. O SOAP define um formato de mensagem, baseado em XML, para troca de dados através da Internet. É um formato simples, fácil de desenvolver e independente de plataforma, linguagem de programação ou sistema operacional (NEWCOMER, 2002). O SOAP utiliza o HTTP como protocolo de transporte, o que torna possível atravessar *firewalls*.

- O WSDL (*Web Services Description Language*) é uma linguagem de descrição de serviços Web. Esta linguagem permite que sejam representadas suas funcionalidades, assim como, parâmetros de entrada e saída, métodos e também o endereço onde se localiza o serviço.

- A publicação dos serviços utiliza o repositório UDDI (*Universal Description, Discovery, and Integration*) . No repositório UDDI são concentrados os serviços Web e é possível realizar uma busca por um determinado serviço. Porém o UDDI não implementa características semânticas, suportando apenas a descoberta sintática.

2.2 SERVIÇOS WEB SEMÂNTICOS

Devido à grande quantidade de informações disponíveis na Internet, sentiu-se a necessidade de adicionar um significado, aos dados, que pudesse ser interpretado por máquinas, não somente humanos. Nesse contexto, surgiu a Web semântica com uma nova proposta de apresentar os dados. A Web semântica não é uma nova Web, e sim uma extensão da atual, onde as informações têm uma estrutura que proporciona a representação semântica dos dados, pos-

sibilitando que pessoas e computadores trabalhem juntos (BERNERS-LEE; HENDLER; LASSILA, 2001).

A Web semântica utiliza uma estrutura hierárquica de conceitos e relacionamentos, esta estrutura é uma ontologia. Através da ontologia é possível realizar associações e inferências sobre os dados de forma automática. Uma ontologia representa o conhecimento de um domínio específico, ou seja, uma ontologia que expressa o domínio de uma universidade não pode ser utilizada no domínio de meios de transporte.

A Web semântica pode ser aplicada em vários contextos, inclusive os serviços Web. O uso da Web semântica incorporada aos serviços Web tornou mais fácil a manipulação desses serviços, principalmente para a automatização das buscas. Com o uso da semântica para descrever serviços Web, a busca automática por um determinado serviço tornou-se mais eficaz. Isso é possível por causa da facilidade que a semântica oferece em realizar inferência e associações sobre um determinado conceito.

Um serviço Web semântico utiliza um documento de descrição semântica, além do WSDL, para descrever suas funcionalidades. É necessário uma linguagem específica, que possibilite representar as ontologias do serviço. Dentre as linguagens criadas, pode-se citar: OWL-S (*Ontology Web Language for Service*) (MARTIN et al., 2004) , WSMO (*Web Service Modeling Ontology*) (LAUSEN; POLLERES; ROMAN, 2005) e o SAWSDL (*Semantic Annotations for WSDL*) (KOPECKY et al., 2007) .

O presente trabalho utiliza a OWL-S como linguagem de descrição de serviço. A OWL-S utiliza um formalismo para representar o conhecimento de uma ontologia, que é a linguagem OWL(*Web Ontology Language*) (BECHHOFER et al., 2004) . Na OWL-S são definidas três ontologias: no *Profile* são descritas as funcionalidades do serviço, é usada para identificar um serviço pelas ferramentas de busca; no *Model* é descrito o comportamento do serviço e como este deve ser invocado; no *Grounding* é descrita a localização do WSDL e a forma de acesso ao serviço, como exemplo o protocolo de comunicação e o formato das mensagens. Esta linguagem será melhor explicada adiante.

2.3 COMPOSIÇÃO DE SERVIÇOS WEB

Muitos serviços, quando isolados, podem não atender aos requisitos do cliente, neste caso, torna-se necessário combinar determinados serviços a fim de que o objetivo seja alcançado. Esta combinação denomina-se composição de serviços, que pode também utilizar características semânticas (CLARO; MACêDO, 2008).

Assim como um serviço simples, uma composição de serviços Web tem parâmetros de entrada, parâmetros de saída, pré-condições e efeitos. Uma composição segue as mesmas características da arquitetura SOA na figura 1. Para um cliente, é indiferente se o serviço que ele vai executar é simples ou composto, o que importa é que o serviço tenha a resposta desejada.

Em uma composição de serviços Web, o parâmetro de saída de um serviço pode ser o parâmetro de entrada de outro, de forma que vários dados são manipulados por diversos serviços até que seja possível chegar ao resultado desejado. A ordem em que os serviços são executados é definida pela composição, pode ser em sequência, em paralelo, depender de uma condição, são várias as possibilidades. Existem diversas linguagens que suportam a composição de serviços, entre elas está a OWL-S.

2.4 OWL-S

As descrições de serviços Web em WSDL não possuem informações semânticas. Estas informações são importantes para a descoberta, invocação e composição automática de serviços. A linguagem OWL-S (*Ontology Web Language for Service*) (MARTIN et al., 2004) é uma linguagem para descrição semântica de serviços Web. A OWL-S é baseada na OWL (*Web Ontology Language*), uma linguagem para descrição de ontologias.

Para representar as informações dos serviços, a OWL-S possui uma ontologia principal que descreve outras três ontologias. A figura 2 mostra como são organizadas essas ontologias.

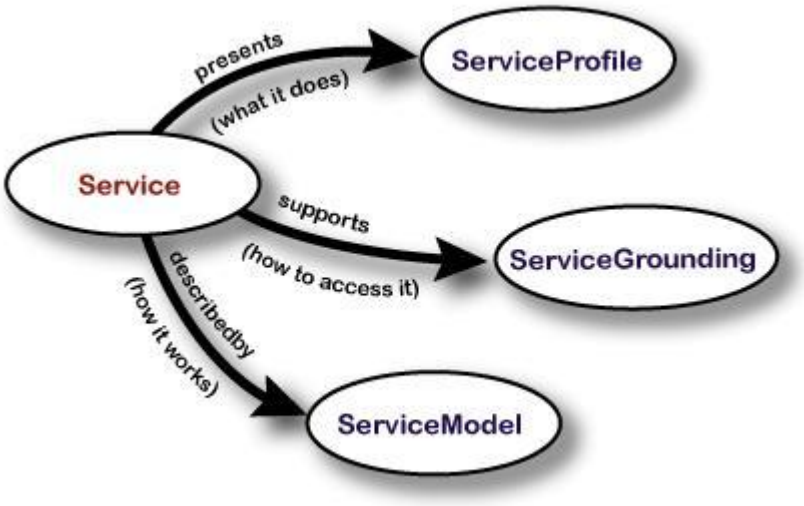

Figura 2: Representação das ontologias OWL-S (MARTIN et al., 2004)

O *Service* é o ponto central da representação de um serviço Web. Todo serviço Web possui uma instância de *Service*. Esta ontologia é descrita através das propriedades *presents*, *descri-*

bedBy e *supports* que indicam as três sub-ontologias *ServiceProfile*, *ServiceModel* e *ServiceGrouding* respectivamente. A listagem 2.1 é um exemplo de descrição da ontologia *Service*.

Listagem 2.1: Ontologia Service

```
<!-- Service -->
<service:Service rdf:ID="getMatriculaAlunoService">
   <service:presents rdf:resource="#getMatriculaAlunoProfile"/>
   <service:describedBy rdf:resource="#getMatriculaAlunoProcess"/>
   <service:supports rdf:resource="#getMatriculaAlunoGrounding"/>
</service:Service>
```

A ontologia *Profile* descreve o que o serviço faz, de forma que facilita a descoberta automática. Nela são descritos os elementos funcionais do serviço, como parâmetros de entrada e saída, pré-condições e efeitos. Pode possuir também informações de contato do provedor do serviço. A listagem 2.2 é um exemplo de descrição da ontologia *Profile*.

Listagem 2.2: Ontologia Profile

```
<!-- Profile -->
<profile:Profile rdf:ID="getMatriculaAlunoProfile">
   <service:isPresentedBy rdf:resource="#getMatriculaAlunoService"/>
   <profile:serviceName xml:lang="en">
      Get Matricula Aluno</profile:serviceName>
   <profile:textDescription xml:lang="en">
      Retorna numero de matricula do aluno</profile:textDescription>
   <profile:hasInput rdf:resource="#cpf"/>
   <profile:hasOutput rdf:resource="#matriculaAluno"/>
</profile:Profile>
```

A ontologia *Model* descreve como usar o serviço, detalhando passo a passo como é o seu comportamento. Um serviço pode ser classificado como atômico, simples ou composto. O serviço atômico é um único serviço que pode ser invocado diretamente através do seu endereço. O serviço simples não pode ser invocado, ele apenas descreve uma forma de utilização do serviço atômico. O serviço composto é formado por vários serviços atômicos, sendo que a ordem de execução dos serviços é definida por controladores de fluxo (MARTIN et al., 2004). A OWL-S possui suporte para os seguintes controladores de fluxo: *Sequence*, *If-Then-Else*, *Any-Order*, *Split*, *SplitJoin*, *Choose* e *Repeat-Until*. A listagem 6.1 é um exemplo de descrição da ontologia *Model*.

Listagem 2.3: Ontologia Model

```
<!-- Model -->
<process:AtomicProcess rdf:ID="getMatriculaAlunoProcess">
   <service:describes rdf:resource="#getMatriculaAlunoService"/>
   <process:hasInput rdf:resource="#cpf"/>
   <process:hasOutput rdf:resource="#matriculaAluno"/>
</process:AtomicProcess>

<process:Input rdf:ID="cpf">
   <process:parameterType rdf:datatype="&xsd;#anyURI">
      http://www.w3.org/2001/XMLSchema#string</process:parameterType>
   <rdfs:label>CPF Aluno</rdfs:label>
</process:Input>

<process:Output rdf:ID="matriculaAluno">
   <process:parameterType rdf:datatype="&xsd;#anyURI">
      http://www.w3.org/2001/XMLSchema#int</process:parameterType>
   <rdfs:label>Numero de Matricula do Aluno</rdfs:label>
</process:Output>
```

Na ontologia *Grounding* está descrito os detalhes de como o serviço pode ser acessado, tais como: protocolo de comunicação, formato de mensagem. Além do mapeamento do serviço no seu respectivo documento WSDL. A listagem 2.4 é um exemplo de descrição da ontologia *Grounding*.

Listagem 2.4: Ontologia Grouding

```
<!-- Grounding -->
<grounding:WsdlGrounding rdf:ID="getMatriculaAlunoGrounding">
   <service:supportedBy rdf:resource="#getMatriculaAlunoService"/>
   <grounding:hasAtomicProcessGrounding
      rdf:resource="#getMatriculaAlunoProcessGrounding"/>
</grounding:WsdlGrounding>

<grounding:WsdlAtomicProcessGrounding
   rdf:ID="getMatriculaAlunoProcessGrounding">
   <grounding:owlsProcess rdf:resource="#getMatriculaAlunoProcess"/>
   <grounding:wsdlDocument rdf:datatype="&xsd;#anyURI">
      &wsdl;</grounding:wsdlDocument>
```

```
    <grounding:wsdlOperation>
        <grounding:WsdlOperationRef>
            <grounding:portType rdf:datatype="&xsd;#anyURI">
                &wsdl;#MatriculaAluno</grounding:portType>
            <grounding:operation rdf:datatype="&xsd;#anyURI">
                &wsdl;#getMatriculaAluno</grounding:operation>
        </grounding:WsdlOperationRef>
    </grounding:wsdlOperation>

    <grounding:wsdlInputMessage rdf:datatype="&xsd;#anyURI">
        &wsdl;#getMatriculaAlunoRequest</grounding:wsdlInputMessage>
    <grounding:wsdlInput>
        <grounding:WsdlInputMessageMap>
            <grounding:owlsParameter rdf:resource="#cpf"/>
            <grounding:wsdlMessagePart rdf:datatype="&xsd;#anyURI">
                &wsdl;#cpf</grounding:wsdlMessagePart>
        </grounding:WsdlInputMessageMap>
    </grounding:wsdlInput>

    <grounding:wsdlOutputMessage rdf:datatype="&xsd;#anyURI">
        &wsdl;#getMatriculaAlunoResponse</grounding:wsdlOutputMessage>
    <grounding:wsdlOutput>
        <grounding:WsdlOutputMessageMap>
            <grounding:owlsParameter rdf:resource="#matriculaAluno"/>
            <grounding:wsdlMessagePart rdf:datatype="&xsd;#anyURI">
                &wsdl;#getMatriculaAlunoReturn</grounding:wsdlMessagePart>
        </grounding:WsdlOutputMessageMap>
    </grounding:wsdlOutput>
</grounding:WsdlAtomicProcessGrounding>
```

3 DESCOBERTA DE SERVIÇOS WEB SEMÂNTICOS

Hoje em dia, muitas empresas, parceiras ou não, compartilham informações na Internet. O mundo dinâmico impõe a integração virtual dessas empresas e parceiros, e eles precisam expor seus serviços através da Internet. Porém, o serviço precisa ser encontrado e geralmente esse é o problema inicial (CLARO, 2004).

Encontrar um serviço Web que atenda às necessidades do cliente não é uma tarefa fácil. Um repositório pode possuir muitos serviços o que torna impossível que essa atividade seja feita manualmente. Por conta disso, existem ferramentas que fazem o processo de descoberta de forma automática.

Dado um repositório de serviços Web e uma requisição de consulta a serviço, encontrar um serviço automaticamente no repositório que atenda aos requisitos da consulta é o problema da descoberta de serviços. Soluções válidas para o problema, devem satisfazer as seguintes condições (KONA; BANSAL; GUPTA, 2007):

- produzir pelo menos os parâmetros de saída do serviço requisitado e satisfazer suas pós-condições;
- usar somente os parâmetros de entrada informados e satisfazer as pré-condições do serviço requisitado;
- produzir os mesmos efeitos que o serviço requisitado.

Para realizar a descoberta, é necessário representar a consulta com uma linguagem específica, onde devem ser informados os parâmetros de entrada e saída do serviço requerido. Para processar todos esses dados e realizar a descoberta, utiliza-se um algoritmo de correspondência também chamado algoritmo de *matching*. Este algoritmo faz a análise da requisição do usuário e compara com todos os serviços existentes no repositório, caso encontre um ou mais serviços compatíveis informa ao usuário.

Caso não seja possível encontrar um serviço compatível, alguns algoritmos oferecem a opção da composição de serviços. Neste caso, os parâmetros de entrada e saída são combinados de forma que se obtenha uma composição de serviços que satisfaça as condições citadas anteriormente.

As informações semânticas são essenciais para os algoritmos de *matching*, pois diminui o problema dos falsos positivos. Falsos positivos ocorrem quando dois parâmetros são considerados sintaticamente similares, mas na verdade são semanticamente diferentes. Isso faz com que os algoritmos que utilizam abordagem semântica sejam mais eficazes.

Alguns serviços podem não atender a todos os requisitos, mas são considerados aceitáveis visto que atendem às necessidades do usuário. Por exemplo, um serviço pode não utilizar todos os parâmetros de entrada fornecidos pelo usuário, porém os parâmetros de saída devem ser iguais. Estes serviços são considerados parcialmente similares. Para resolver tal problema, existe um conceito chamado de grau de correspondência (ou *Degree of Match* - DoM) . O DoM é um número que é calculado utilizando-se algumas métricas e representa o grau de similaridade entre os serviços.

A seguir, serão explicados alguns algoritmos estudados e o que foi utilizado no presente trabalho.

3.1 CORRESPONDÊNCIA POR CAPACIDADES SEMÂNTICAS

Um algoritmo de descoberta utilizando capacidades semânticas foi apresentado em (PAOLUCCI et al., 2002). Este algoritmo é baseado na ontologia *Profile* do OWL-S, aproveitando as informações semânticas que a linguagem oferece. Isso torna possível fazer inferências em cima da hierarquia de conceitos semânticos o que facilita a descoberta de serviços mesmo que sejam sintaticamente diferentes.

A partir de uma requisição do usuário, o algoritmo deve retornar os serviços que são suficientemente similares à requisição. Porém, o conceito de suficientemente similar é muito abstrato, por esta razão o trabalho de Paolucci utiliza o conceito de grau de similaridade que tem o mesmo objetivo do DoM citado anteriormente.

O algoritmo apresentado compara todos os parâmetros de entrada e saída da requisição, com os parâmetros de todos os serviços publicados no repositório. A correspondência entre os parâmetros é baseada na ontologia de domínio a qual os parâmetros pertencem. Considere a

figura 3, que representa uma ontologia de veículo, o grau de similaridade é calculado de acordo com as seguintes regras:

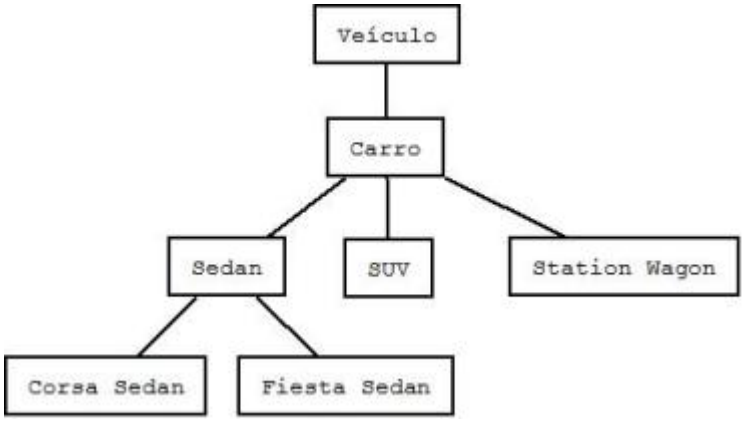

Figura 3: Parte de uma ontologia de veículo

Exact: se os parâmetros são iguais ou se o parâmetro requisitado é subclasse direta do parâmetro do serviço. Por exemplo, de acordo com a ontologia representada na figura 3, se ambos os parâmetros requisitado e o do serviço são do tipo *Station Wagon*, ou, o parâmetro requisitado é *Station Wagon* e o do serviço é do tipo Carro.

Plug in: quando um parâmetro requisitado é subclasse do parâmetro do serviço, exceto no caso que corresponde ao *exact*. Por exemplo, de acordo com a ontologia representada na figura 3, é requisitado um parâmetro do tipo *Corsa Sedan* e o parâmetro do serviço é do tipo Carro.

Subsumes: ao contrário do *plug in*, é quando um parâmetro do serviço é subclasse do parâmetro requisitado. Por exemplo, de acordo com a ontologia representada na figura 3, é requisitado um parâmetro do tipo Carro e o parâmetro do serviço é do tipo *Fiesta Sedan*.

Fail: quando não há relação alguma entre os parâmetros.

3.2 CORRESPONDÊNCIA HÍBRIDA

Observou-se que o algoritmo de correspondência semântica não é eficaz em todos os casos. Podem existir parâmetros que não são semanticamente similares, mas que são úteis para o usuário. Baseando-se nessa falha, a abordagem híbrida compara os parâmetros sintaticamente, além da comparação semântica.

Um algoritmo proposto em (KLUSCH; FRIES; SYCARA, 2006) utiliza essa abordagem. Neste algoritmo, além dos graus de similaridade *Exact*, *Plug in* e *Subsumes*, que consideram apenas a relação semântica entre os parâmetros, existem também os filtros *Subsumed by* e *Nearest-*

Neighbor, que consideram o relacionamento semântico e sintático juntos obtido através de um cálculo de similaridade.

O algoritmo escolhido para ser utilizado no presente trabalho tem como base essa abordagem, visto que é mais eficaz que a correspondência semântica apenas.

3.3 OWL-S DISCOVERY

O OWL-S Discovery é uma ferramenta de descoberta semântica proposta por (AMORIM, 2009). Utiliza um algoritmo híbrido, com uma etapa semântica funcional e uma semântica descritiva.

A etapa semântica funcional utiliza os mesmos graus de similaridades, também chamados de filtros, indicados na seção 3.1, que são *Exact*, *Plug in* e *Subsumes*. Além deles foi criado também o filtro *Sibling* baseado no trabalho de (SAMPER et al., 2008). Duas classes são classificadas como *Sibling* se elas tiverem o mesmo pai. Por exemplo na figura 3, as classes SUV e *Station Wagon* são classificadas como *Sibling*.

O funcionamento do algoritmo necessita de um diretório com as descrições dos serviços Web, onde é feita a busca, um meio de acesso as ontologias que são referenciadas nas descrições dos serviços, e uma requisição de consulta do usuário. Com isso, é feita uma correspondência entre todos os parâmetros dos serviços do repositório com os parâmetros requisitados pelo usuário. Cada parâmetro terá seu grau de similaridade, e o menor deles irá representar o grau de similaridade do serviço em questão.

A etapa semântica descritiva, baseada no trabalho de (LOPES et al., 2006), faz uma comparação das classes e uma consulta ao dicionário de sinônimos, que deve ser fornecido pelo usuário. O algoritmo utiliza uma função que calcula a similaridade entre duas classes. Esta função é uma soma ponderada entre a similaridade estrutural e a similaridade básica.

A similaridade básica é calculada através de uma função ψ que retorna 1.0 (um) se as classes são iguais, 0.0 (zero) se as classes são diferentes e 0.5 (meio) se as classes tiverem o mesmo significado de acordo com o dicionário.

A similaridade estrutural se baseia no fato de que duas classes são tão semanticamente similares quanto os seus vizinhos estruturais. Os vizinhos estruturais de uma classe C são divididos em quatro tipos:

- Ancestrais: são as classes que são pai da classe C, desde o pai direto até o elemento raiz

da hierarquia.

- Irmãos: são as classes que possuem o mesmo pai direto que a classe C.
- Filhos diretos: são as classes que são descendentes direto da classe C.
- Folhas: são as classes folhas que tem como pai, mesmo que indiretamente, a classe C.

O cálculo da similaridade estrutural é mais custoso computacionalmente, se comparado ao cálculo da similaridade básica, pois é preciso calcular os quatro tipos acima citados para cada parâmetro dos serviços.

4 TRABALHOS RELACIONADOS

Nesta seção, serão apresentados os trabalhos relacionados a este projeto, os quais foram comparados com o *plugin OWL-S Composer 2.0*.

4.1 CMU'S OWL-S DEVELOPMENT ENVIRONMENT

A ferramenta *CMU's OWL-S Development Environment* (CODE) (CMU, 2005) é um *plugin* para o Eclipse que oferece suporte a todo o processo de desenvolvimento de serviços Web usando OWL-S (SRINIVASAN; PAOLUCCI; SYCARA, 2005). Através do CODE, é possível gerar o documento WSDL a partir de código Java, gerar o OWL-S a partir do WSDL além de publicar o serviço em repositórios UDDI. Essa ferramenta provê, com isso, um ambiente de desenvolvimento unificado para a criação de serviços OWL-S.

Além disso, esta ferramenta também oferece o suporte para o cliente de serviços Web, como a descoberta de serviços em um repositório UDDI. O usuário pode fazer uma consulta e obter o serviço requerido ou similares. Entretanto, esta descoberta é através da análise sintática, evidenciando uma limitação da ferramenta. É possível também, gerar o código em Java que faz a chamada aos serviços Web.

A ferramenta possui quatro editores que suportam os diferentes fragmentos do documento OWL-S, *profile*, *process model*, *grounding* e *service*. A figura 4 demonstra a interface do *profile editor* onde é possível editar o OWL-S através de formulário. Outra opção ao usuário é editar o OWL-S diretamente no código, apesar de não ser o objetivo deste *plugin*.

Um aspecto positivo da ferramenta é a integração com o Eclipse, um ambiente de desenvolvimento que tem suporte a serviços Web. Isso facilita o desenvolvimento pois tudo pode ser feito em um ambiente único. Porém, a ferramenta falha na representação gráfica das composições de serviços, a qual é feita utilizando árvores e formulários. O que torna difícil uma visualização mais completa e intuitiva para o usuário.

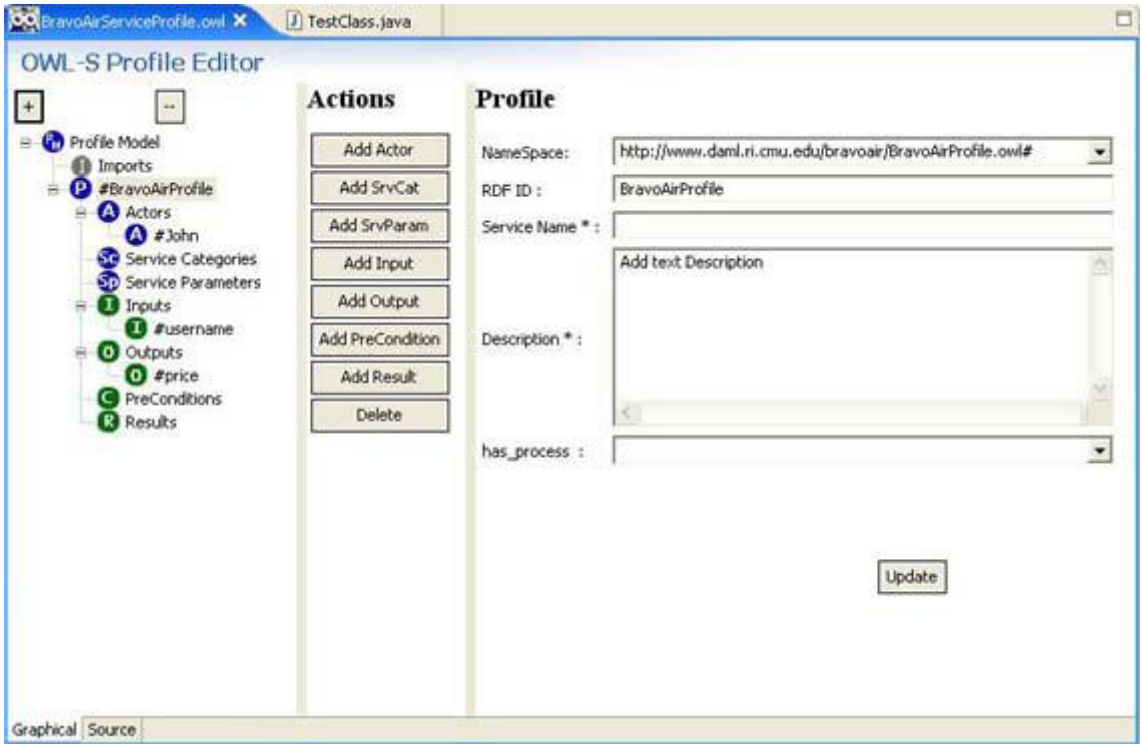

Figura 4: Layout principal do CODE (SRINIVASAN; PAOLUCCI; SYCARA, 2005).

4.2 DISCOVERY AND COMPOSITION ENGINE

Esta ferramenta realiza a descoberta de serviços Web semânticos. Se o serviço não for encontrado, são procurados outros serviços e é gerada uma composição de forma a atender os requisitos do usuário. O fluxo da composição é determinado automaticamente sem a necessidade de intervenção do usuário. É utilizada a linguagem USDL (*Universal Service-Semantics Description Language*) para descrição semântica dos serviços Web (KONA; BANSAL; GUPTA, 2007).

O algoritmo implementado cria a composição em vários passos, onde são analisadas as entradas, saídas, pré-condições e efeitos de cada serviço. O objetivo é encontrar uma solução que seja um grafo acíclico de serviços, que possam ser compostos de forma a produzir o serviço requisitado pelo usuário. Isto é feito filtrando os serviços que não são úteis, durante várias etapas do processo. O algoritmo inicia com a lista dos parâmetros de entrada E, fornecidos pelo usuário. São selecionados todos os serviços do repositório que possuem como entrada um subconjunto de E. Na próxima etapa, é adicionada a lista de parâmetros de entrada E, todos os parâmetros de saída dos serviços encontrados na primeira etapa. Com isso, o processo é repetido até que na lista de parâmetros de saída, estejam todos os parâmetros requisitados pelo usuário.

A descoberta dos serviços é baseada nas descrições semânticas e busca sempre por ótimas soluções baseadas em critérios como custo dos serviços e quantidade de serviços em uma composição. As principais desvantagens deste trabalho é que este ainda não está integrado com um ambiente de desenvolvimento. Além disso, utiliza a linguagem USDL que não é tão difundida como a OWL-S.

4.3 OWL-S COMPOSER 1.0

O *OWL-S Composer* é um plugin desenvolvido para compor serviços Web semânticos através da plataforma Eclipse. Este plugin tem como principais características a utilização de um ambiente visual para gerar composições de serviços, a integração com outros *plugins* do Eclipse, como o WTP (*Web Tool Plugin*) (WTP, 2009) , minimizando a curva de aprendizagem e a facilidade de manipular as composições além de gerar o código da composição de uma maneira sintática e semanticamente correta (FONSECA; CLARO; LOPES, 2009).

Por ser um *plugin* do Eclipse, o *OWL-S Composer* tem todo o suporte necessário de um ambiente de desenvolvimento integrado para desenvolvimento e manipulação de serviços Web. Isto facilita o trabalho do desenvolvedor, que não vai precisar utilizar várias ferramentas ao mesmo tempo.

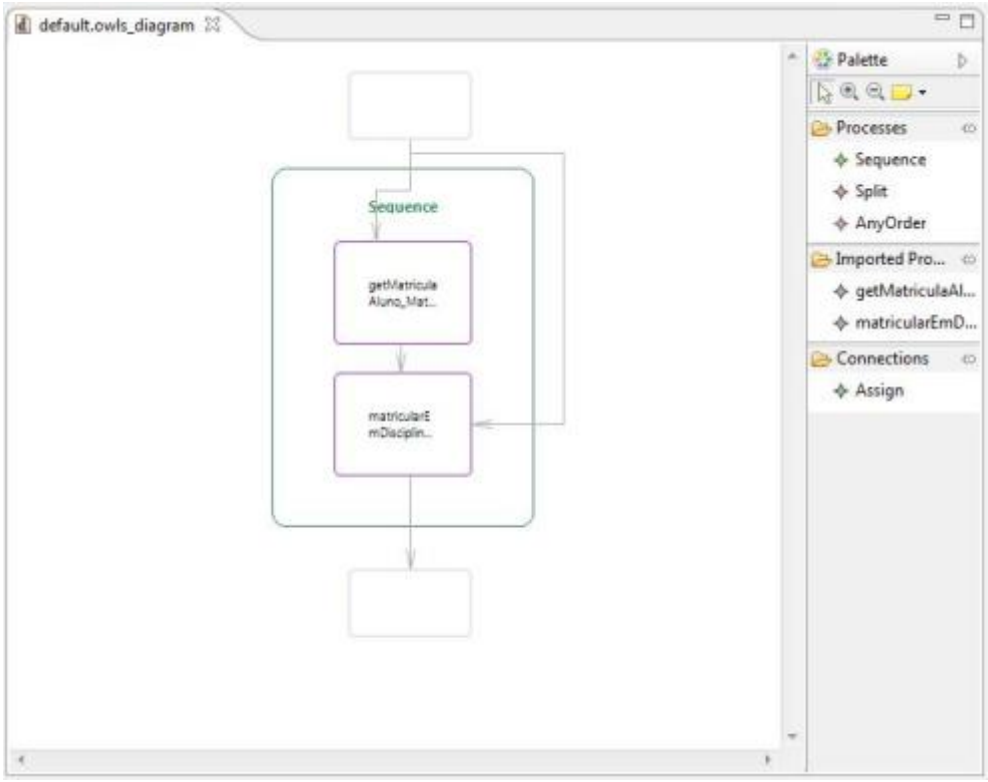

Figura 5: Exemplo de uma diagrama criado pelo OWL-S Composer 0.1

O *OWL-S Composer* possui um editor de diagramas onde é possível modelar graficamente a composição dos serviços previamente escolhidos pelo usuário. Na figura 5, está ilustrado o editor de composições do *plugin*, no qual foi gerada uma composição do tipo *Sequence*, formada por dois serviços: *getMatriculaAluno* e *matricularEmDisciplina*. O *OWL-S Composer* possui suporte a apenas três estruturas de controle, que são: *Sequence*, *Split* e *Any-Order*.

4.3.1 ARQUITETURA

Este *plugin* foi desenvolvimento com o suporte de outros *plugins* da plataforma Eclipse, são eles: EMF (*Eclipse Modeling Framework Project*) (EMF, 2009) , GEF (*Graphical Editing Framework*) (GEF, 2009) , GMF (*Graphical Modeling Framework*) (GMF, 2009) , JET (*Java Emitter Templates*) (JET, 2009) . Estes plugins são utilizados na criação e edição de diagramas e na geração do código referente à composição dos serviços. A seguir, serão explicados cada um desses *plugins*.

- **EMF**: *Eclipse Modeling Framework Project* é um *framework* que oferece suporte a modelagem e geração de código para ferramentas e outras aplicações baseadas em modelos de classes simples. Com o EMF, é possível gerar código Java a partir de diagramas UML ou XML Schema, ou ao contrário através de engenharia reversa. O EMF foi utilizado no *OWL-S Composer* para modelar a linguagem OWL-S baseada na especificação da versão 1.1.

- **GEF**: *Graphical Editing Framework* permite a criação de elementos gráficos a partir de modelos de aplicações existentes. O GEF implementa a arquitetura MVC (*Model-View-Controller*), o que facilita a integração da parte gráfica com o modelo. O *OWL-S Composer* utiliza o GEF para criar os elementos gráficos do diagrama que representa a composição de serviços.

- **GMF**: a ferramenta *Graphical Modeling Framework* faz a integração entre o EMF e o GEF. Através do GMF, é possível fazer o mapeamento dos elementos gráficos no modelo de classes. Isso é feito de forma visual, facilitando o trabalho do desenvolvedor. Com o GEF, o *OWL-S Composer* faz o mapeamento dos elementos da OWL-S nos objetos gráficos que irão compor o diagrama.

- **JET**: *Java Emitter Templates* é uma ferramenta para geração de código a partir de *templates*, os quais são desenvolvidos usando linguagem JSP (*Java Server Pages*). O JET torna a geração automática de código uma tarefa simples. O *plugin OWL-S Composer*

utiliza o JET para gerar o código correspondente a composição de serviços representada pelo diagrama.

Além dos *plugins* citados acima, o *OWL-S Composer 1.0* utiliza a OWL-S API (MINDSWAP, 2007) e a ferramenta JAX-SA (BABIK, 2008). A OWL-S API provê todo o suporte para leitura, execução e escrita de descrições OWL-S através de código Java. Além disso, a OWL-S API faz o mapeamento das descrições semânticas dos serviços OWL-S em elementos do *OWL-S Composer*.

JAX-SA é uma ferramenta desenvolvida para o tratamento de anotações semânticas de serviços Web. No *OWL-S Composer 1.0* esta ferramenta foi utilizada para transformar arquivos WSDL em OWL-S. Esta é uma etapa opcional, pois o usuário pode já possuir as descrições dos serviços Web em formato OWL-S. A figura 6 representa a arquitetura sob a qual está desenvolvida o *plugin OWL-S Composer* na versão 1.0.

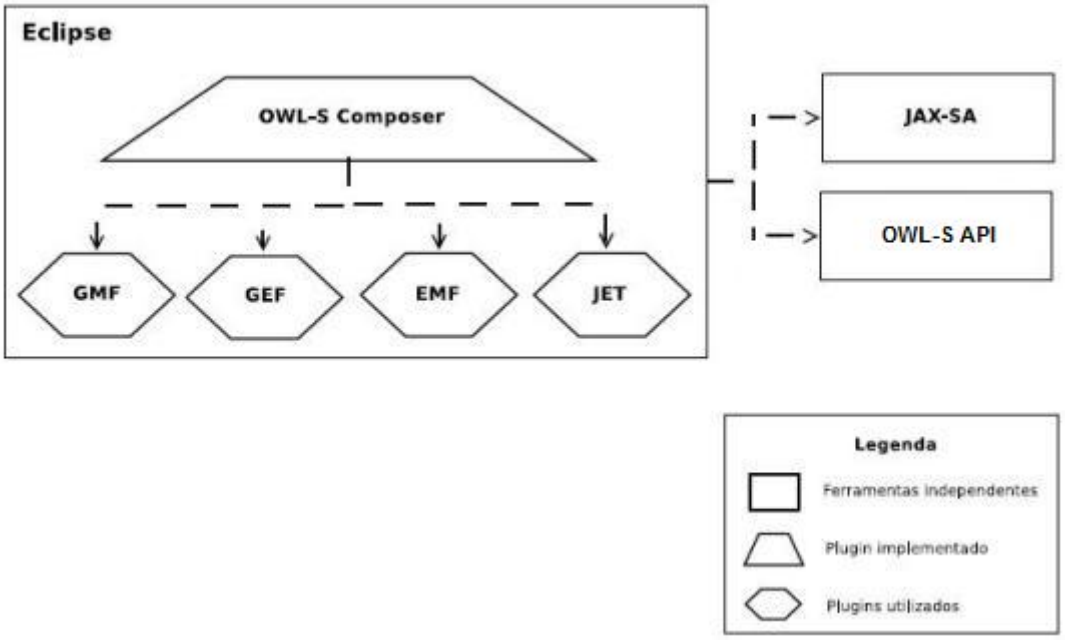

Figura 6: Arquitetura do OWL-S Composer 1.0(FONSECA; CLARO; LOPES, 2009)

4.3.2 LIMITAÇÕES DO OWLS-COMPOSER 1.0

Este plugin não pode ser considerado completo, pois não abrange todo o ciclo de vida de uma composição OWL-S. O *plugin* não implementa todos os controladores de fluxo da especificação OWL-S, apenas o *Sequence*, *Split* e *AnyOrder*. Isto ocorre devido a uma limitação da ferramenta utilizada, OWL-S API, que não possui suporte para os outros controladores de

fluxo. Além disso, o *plugin* não realiza a implantação do serviço em um servidor e não faz a execução do mesmo.

Outra funcionalidade que não foi implementada no *OWL-S Composer 1.0* é a descoberta de serviços Web semânticos que é um passo importante para a composição automática. Neste sentido, o presente trabalho tem como principal objetivo incorporar ao *OWL-S Composer* a busca por composições semanticamente similares, de forma a sanar esta limitação do *plugin*.

5 DESENVOLVIMENTO DO OWL-S COMPOSER 2.0

Este capítulo apresenta o passo a passo do desenvolvido da versão 2.0 do *OWL-S Composer*, que fornece ao usuário a possibilidade de realizar a descoberta de composições de serviços Web, semanticamente similares, de forma visual. Desta maneira, o usuário possui a opção de utilizar outras composições, sem ficar limitado a primeira.

5.1 HISTÓRICO

O *plugin OWL-S Composer 1.0* foi desenvolvido por (FONSECA; CLARO; LOPES, 2009), com o objetivo de oferecer suporte ao desenvolvimento de composições de serviços Web semânticos de forma visual, através de diagramas.

A ferramenta OWL-S Discovery, desenvolvida por (AMORIM, 2009), realiza descoberta semântica de serviços Web atômicos, ou seja, não realiza descoberta de composições de serviços Web.

Este trabalho, uniu as duas ferramentas citadas acima, resultando no *plugin OWL-S Composer 2.0* que realiza a descoberta de serviços Web semânticos incluindo serviços compostos. Além da descoberta, o *plugin* apresenta as composições semanticamente similares de forma visual.

5.2 AMBIENTE

O *OWL-S Composer 2.0* foi desenvolvido sob a versão 3.4.1 do Eclipse, utilizando os *plugins* EMF 2.4.1, GMF 1.1.0, GEF 3.4.1 e JET 1.1.0, que já foram apresentados na seção 4.3. Para executar o *plugin OWL-S Composer* é necessário o mesmo ambiente, sendo que pode haver incompatibilidade com outras versões destas ferramentas.

Como ferramentas independentes da plataforma Eclipse, foram utilizadas a JAX-SA versão 1.6, e a OWL-S Api versão 1.1.0, também apresentadas na seção 4.3.

5.3 PROJETO ESTRUTURAL

Esta seção expõe as principais decisões sobre este trabalho, em tempo de projeto. Estas decisões foram baseadas no estudo realizado sobre os algoritmos de descoberta semântica apresentados no capítulo 3, e sobre as linguagens de descrição de serviços Web semânticos presentes no capítulo 2.

5.3.1 ALGORITMO UTILIZADO

Dentre os algoritmos de descoberta semântica descritos no capítulo 3, os que possuem correspondência híbrida apresentaram ser mais eficazes na obtenção dos resultados. Por este motivo, ao escolher o algoritmo para este trabalho, foi dada preferência a utilização de um algoritmo que possuísse tanto a etapa semântica como a etapa sintática no seu processo de descoberta. Os algoritmos estudados, com essa característica, foram os algoritmos implementados nas ferramentas OWLS-MX e OWL-S *Discovery*.

O critério para escolher entre estes dois algoritmos foi a análise da etapa semântica funcional, visto que é a etapa mais significativa para a identificação de resultados precisos. Nesta etapa, o OWL-S *Discovery* implementa um filtro a mais em comparação ao OWLS-MX, oferecendo mais uma opção ao usuário de encontrar serviços que podem ser importantes. Além disso, o OWL-S *Discovery* é flexível, tornando optativa a utilização apenas da etapa semântica funcional ou apenas a etapa semântica descritiva. Por estes motivos, o OWL-S *Discovery* foi escolhido para ser utilizado no presente trabalho.

O OWL-S *Discovery* utiliza um algoritmo híbrido para descoberta semântica. Porém, a parte semântica descritiva baseia-se em um dicionário de sinônimos, a partir do qual são feitas associações sintáticas. Este dicionário não utiliza um formato padronizado, o que torna difícil o trabalho do usuário. Por conta disso, decidiu-se utilizar apenas a parte semântica funcional do algoritmo. Isto não influenciou na escolha do algoritmo pois esta etapa pode ser facilmente estendida em trabalhos futuros, assim que o dicionário estiver padronizado.

5.3.2 ARQUITETURA

O *OWL-S Composer 2.0* não sofreu mudanças na arquitetura quanto as ferramentas utilizadas, exceto pela inclusão da *OWL-S Discovery*. Esta foi a ferramenta escolhida para realizar a descoberta de serviços semanticamente similares. Para cada serviço de uma composição, produzida pelo usuário, o algoritmo realiza uma busca no diretório indicado para encontrar serviços similares. O algoritmo não retorna composições de serviços, no caso de não encontrar um serviço compatível.

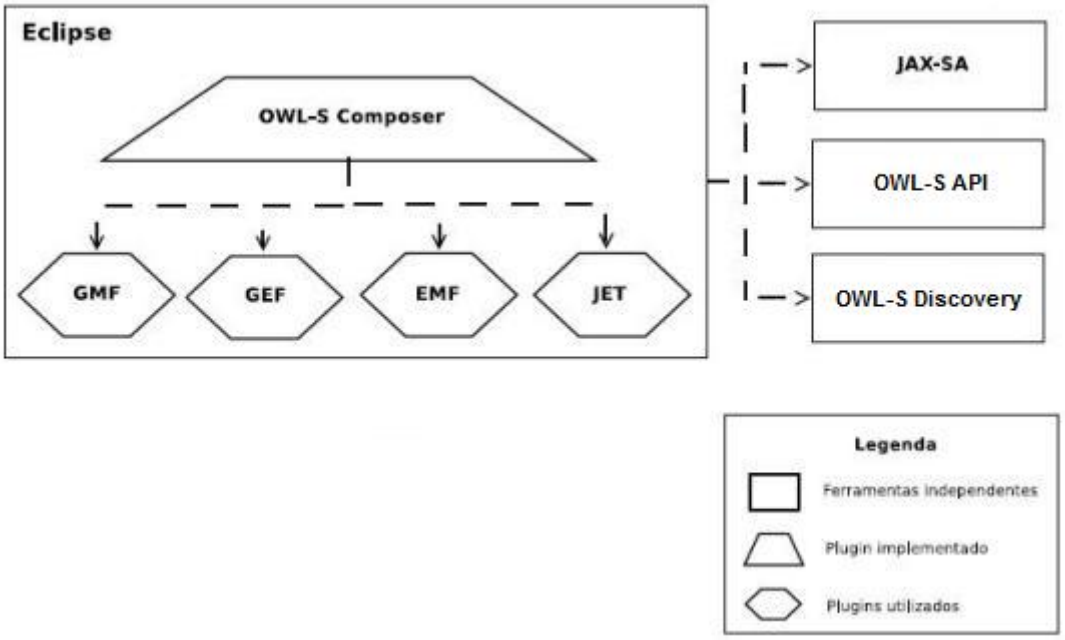

Figura 7: Arquitetura do OWL-S Composer 2.0.

A figura 7 demonstra a arquitetura do *plugin OWL-S Composer 2.0*. O *OWL-S Composer* foi desenvolvido para o Eclipse e depende de outros *plugins*, o EMF, o GMF, o GEF e o JET. O GMF é utilizado da mesma forma que a versão anterior, para gerar os diagramas semanticamente similares. Não foram feitas modificações no meta-modelo do EMF, nem do GEF. O código OWL-S pode ser gerado também para as composições similares. Esta geração automática de código continua sendo realizado pelo JET, sendo que não houve alteração no *template*. Além disso, são utilizadas as ferramentas independentes *JAX-SA*, *OWL-S API* e *OWL-S Discovery*. O *JAX-SA* não foi necessária para implementar a nova funcionalidade. Porém foi preciso a *OWL-S API* para interpretar os serviços retornados pelo *OWL-S Discovery* e mapear suas funcionalidades no meta-modelo.

5.4 IMPLEMENTAÇÃO

A implementação da descoberta semântica foi possível a partir da incorporação do *OWL-S Discovery* no *plugin*.

Para cada serviço da composição, é realizada uma busca por serviços semanticamente similares utilizando o *OWL-S Discovery*. O resultado desta busca depende dos filtros indicados pelo usuário. Finalizada a busca, cada serviço da composição tem uma lista de serviços similares. Esta lista é ordenada de acordo com o grau de similaridade de cada um. Em seguida, os serviços são combinados em novas composições, na ordem das listas, respeitando o fluxo de execução da composição original. Desta forma, o resultado sempre será composições com o maior grau de similaridade dentre os filtros indicados pelo usuário. O grau de similaridade semântica que representa a composição é o menor dos graus de similaridades dos serviços que fazem parte da composição. O *plugin* limita-se a retornar no máximo três composições semanticamente similares.

Ao criar uma composição, o usuário precisa informar os parâmetros de entrada e saída. Estes parâmetros devem ser de um tipo de dado representado por uma ontologia, para que seja possível efetuar a descoberta semântica. Na versão 1.0 do *OWL-S Composer*, é possível apenas utilizar os tipos de dados primitivos definidos pelo XML Schema (SPERBERG-MCQUEEN; THOMPSON, 2000), como por exemplo: *byte*, *boolean*, *date*, *decimal*, *double*, *float*, *int*. Por este motivo acrescentou-se, no *OWL-S Composer 2.0*, a opção para representar os parâmetros através de ontologias. Na figura 8 é exibida a janela onde é realizada a criação da composição. Nesta janela o usuário deve informar o nome, *namespace* e os parâmetros de entrada e saída da composição. Caso o usuário não queira realizar descoberta semântica, não é obrigatório o uso de ontologias como tipo de dado, por este motivo foi preservada a opção do uso dos tipos primitivos.

A parte gráfica dos diagramas, com as composições semanticamente similares, foram gerados através do GMF. Aliado ao diagrama é gerado o meta-modelo contendo as informações semânticas do serviço. É utilizada a OWL-S Api para ler os serviços atômicos encontrados e realizar o mapeamento no meta-modelo.

5.5 ESTUDO DE CASO

Esta seção mostra um estudo de caso realizado para demonstrar a descoberta de composições semanticamente similares através do *plugin OWL-S Composer* 2.0. Os serviços utilizados

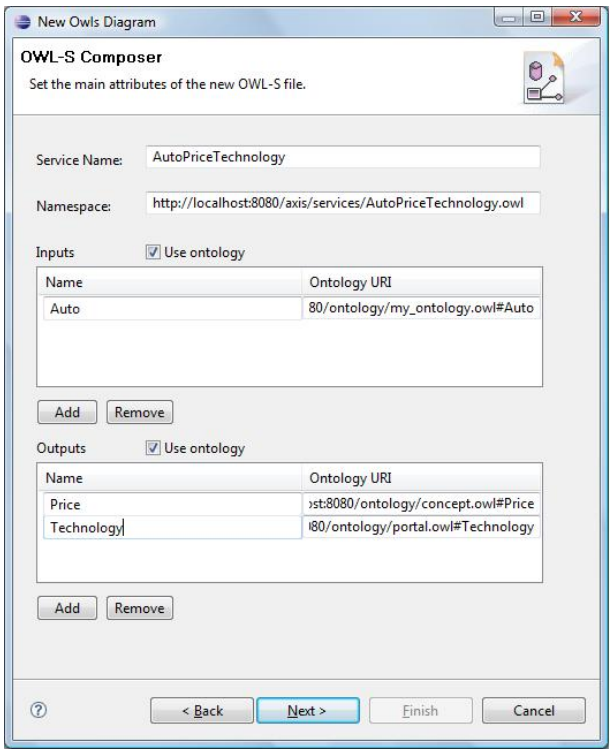

Figura 8: Janela de criação de diagrama de composição do *OWL-S Composer 2.0*.

neste exemplo foram obtidos do *Online Portal for Semantic Services* (KRUG; KüSTER; RUHMER, 2009), um portal colaborativo sobre serviços Web semânticos no qual são compartilhados diversos serviços para testes, com o objetivo de oferecer suporte ao desenvolvimento de trabalhos científicos.

5.5.1 SERVIÇOS E ONTOLOGIAS

Neste estudo de caso, foi gerado o serviço composto *AutoPriceTechnology* que possui como entrada um parâmetro do tipo *Auto* e como saída dois parâmetros, um *Price* e um *Technology*. Estes parâmetros são especificados através de ontologias. A seguir são explicadas partes destas ontologias.

A hierarquia da classe *Auto* está representada pela figura 9. *Auto* tem como classe filha um *Car* e como pai um *WheeledVehicle*.

A hierarquia da classe *Technology* está representada pela figura 10. *Technology* é um *Thing* e possui como filhos *InformationTechnology* e *ComputingTechnology*.

A hierarquia da classe *Price* está representada pela figura 11. *Price* é um *UntangibleObjects* e possui como filhos *MaxPrice*, *RecommendedPrice*, *TaxFreePrice* e *TaxedPrice*.

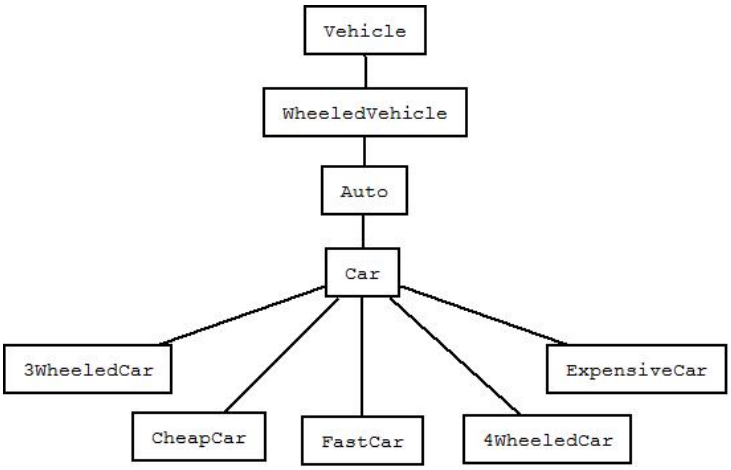

Figura 9: Ontologia de Car

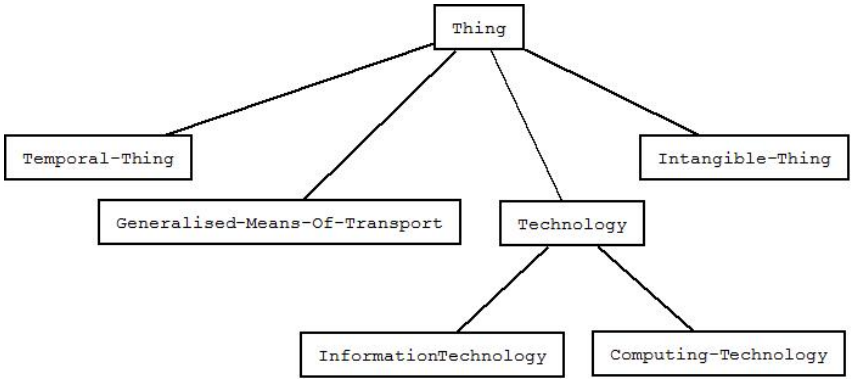

Figura 10: Ontologia de Technology

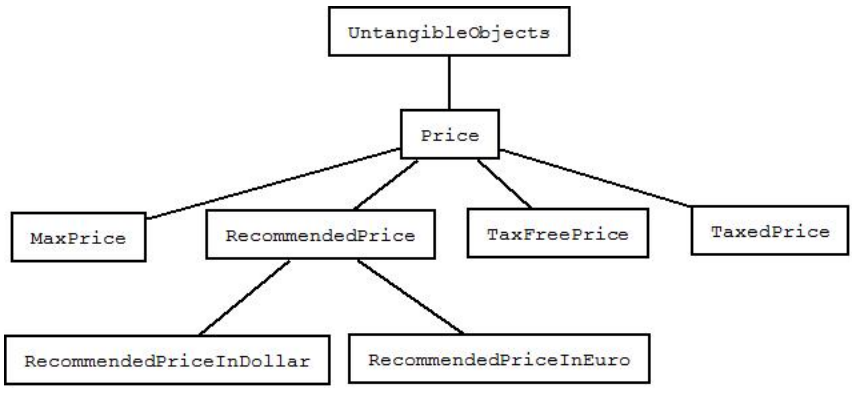

Figura 11: Ontologia de Price

As ontologias são essenciais para a descoberta semântica, pois a partir delas são feitas as inferências entre os parâmetros do serviço.

O serviço *AutoPriceTechnology* foi modelado com o controlador de fluxo *Any-Order* e é composto por dois serviços atômicos. O serviço *getAutoTechnology_AutoTechnology*, que possui como entrada um *Auto* e como saída um *Technology*, e o serviço *getCarPrice_CarPrice* que também possui como entrada um *Auto* e como saída um *Price*. Por ser *Any-Order*, os serviços poderão ser executados em qualquer ordem, não havendo uma dependência entre os parâmetros de um serviço para o outro. Na figura 12, está ilustrada a representação desta composição, gerada através do *OWL-S Composer 2.0*.

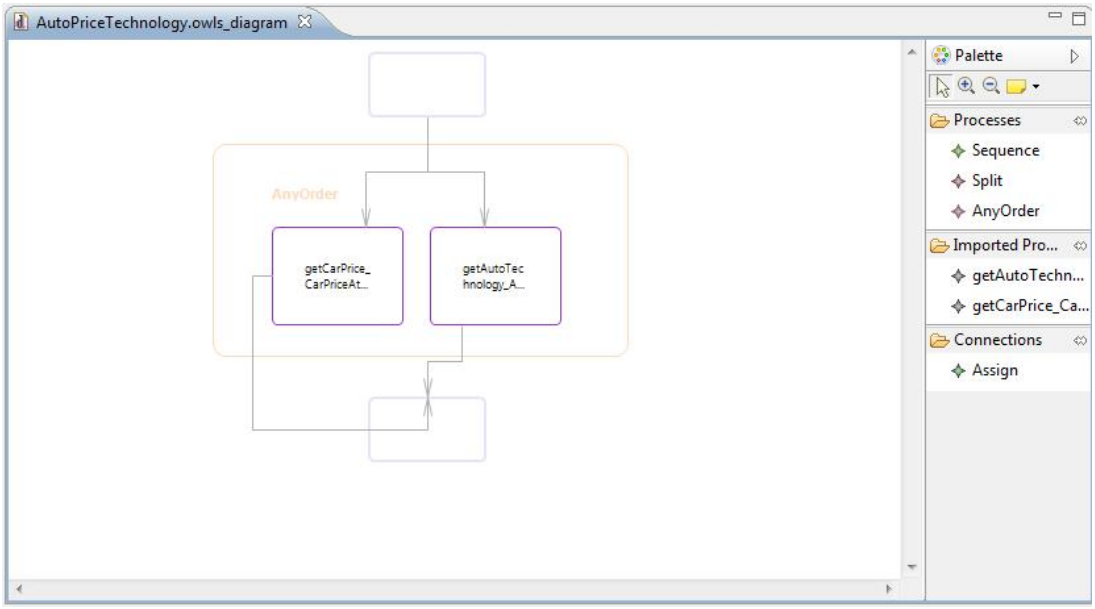

Figura 12: Representação do serviço composto *AutoPriceTechnology* através do *OWL-S Composer 2.0*.

5.5.2 DESCOBERTA

Neste estudo de caso, será realizada a descoberta de serviços semanticamente similares ao *AutoPriceTechnology*, explicado na seção anterior.

A descoberta semântica depende da existência de um diretório local com arquivos que descrevem serviços Web semânticos, na linguagem OWL-S. É neste diretório que será realizada a pesquisa por serviços semanticamente similares. Devido a limitações do *OWL-S Discovery*, não é possível efetuar busca em repositório remoto.

O usuário possui a opção de definir o grau de similaridade semântica que deverá ter o serviço retornado pela consulta. Os filtros disponíveis são: *Exact*, *Plugin*, *Subsumes* e *Sibling*.

Estes dados devem ser informados pelo usuário quando ele solicita a descoberta de serviços similares. Na figura 13 pode-se visualizar a janela na qual são inseridos esses dados, além do caminho do diretório local.

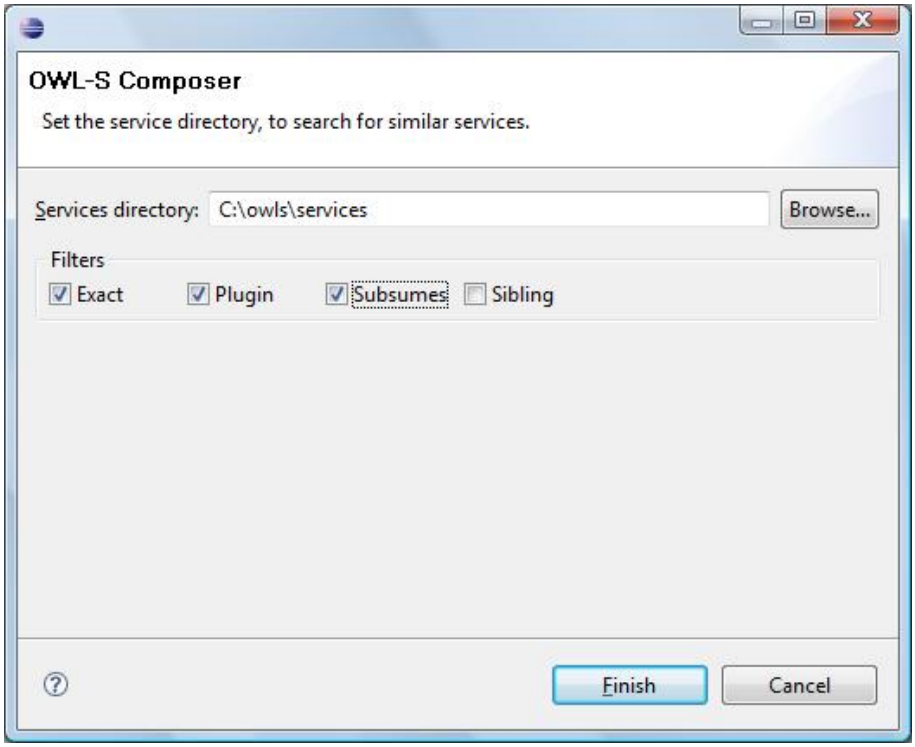

Figura 13: Janela de descoberta de serviços semanticamente similares, do *OWL-S Composer 2.0*

Neste exemplo, foram encontrados dois serviços compostos, similares ao *AutoPriceTechnology*. O resultado pode ser visto na figura 14. O primeiro serviço possui grau de similaridade *Exact*, pois ambos os serviços que o compõe tem o mesmo nível de similaridade com relação ao serviço procurado. Já o segundo serviço, possui grau de similaridade *Subsumes*, pois os serviços que o compõem, *AutoRecommendedpriceAtomicProcess* e *Car_TechnologyAtomicProcess* são respectivamente *Subsumes* e *Exact*, o menor grau prevaleceu.

Um diagrama é criado para cada composição encontrada. Este diagrama possui o mesmo fluxo de controle da composição feita pelo usuário. Os diagramas gerados pelo *OWL-S Composer 2.0* podem ser visualizados nas figuras 15 e 16.

5.6 LIMITAÇÕES DO OWL-S COMPOSER 2.0

O *OWL-S Composer* 2.0 resolveu o problema da descoberta de serviços semanticamente similares, presente na versão anterior do *plugin*. Contudo, ainda existem limitações na ferramenta e estas podem ser exploradas como trabalhos futuros.

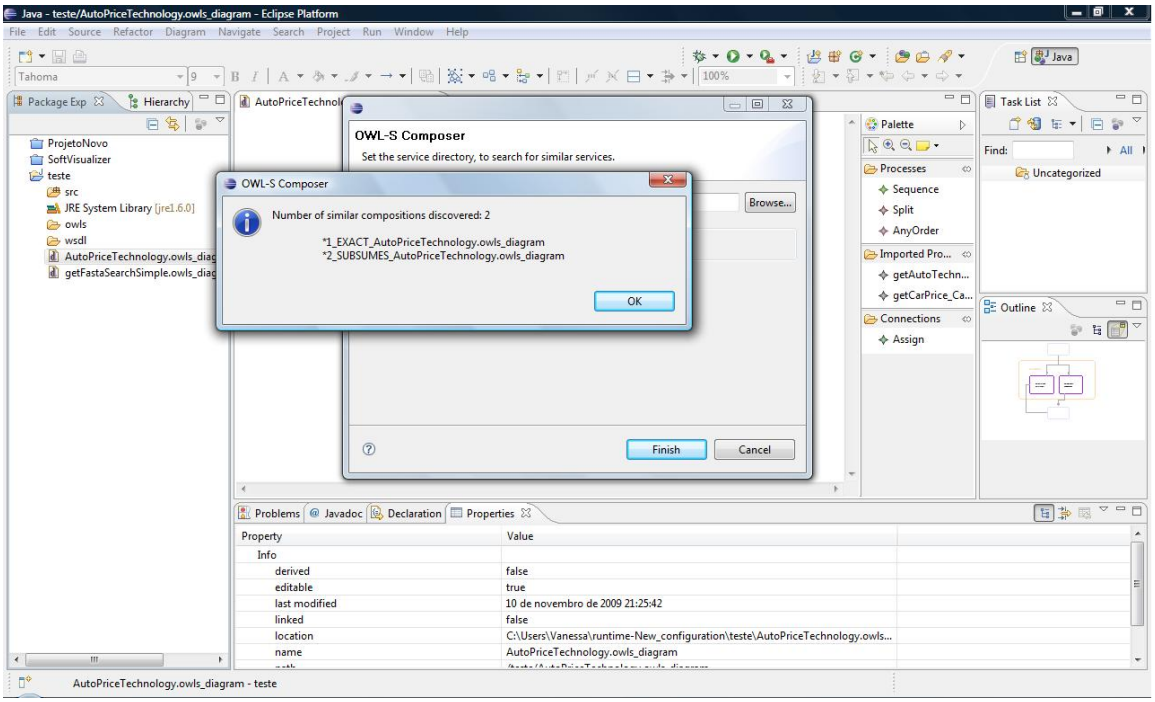

Figura 14: Resultado da descoberta de serviços semanticamente similares, através do *OWL-S Composer 2.0*

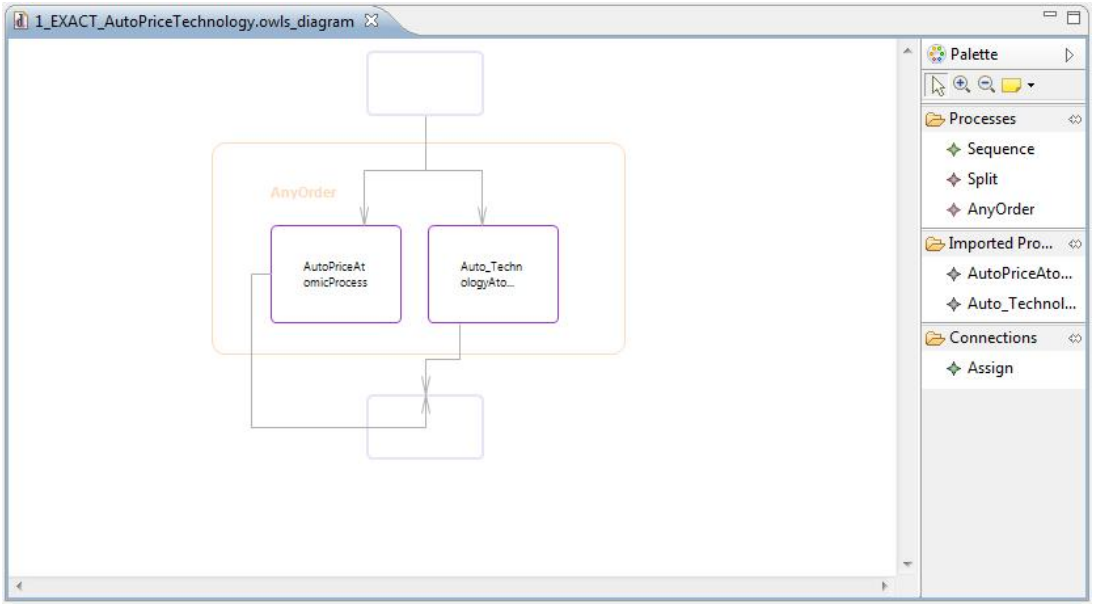

Figura 15: Serviço encontrado com grau de similaridade *Exact*, através do *OWL-S Composer 2.0*.

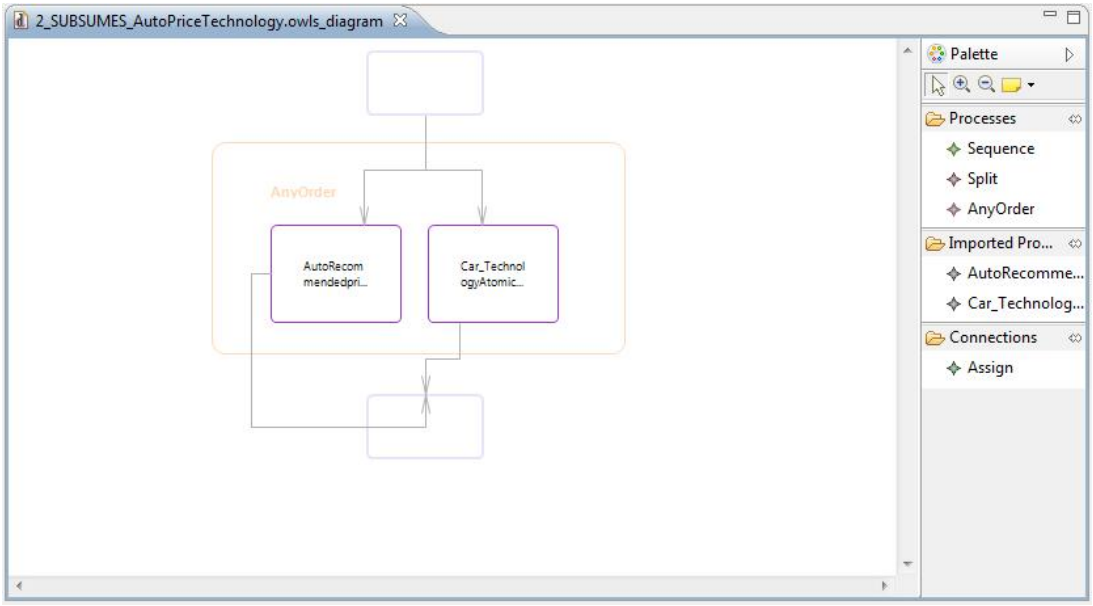

Figura 16: Serviço encontrado com grau de similaridade *Subsumes*, através do *OWL-S Composer 2.0*.

5.6.1 EXTENSÕES NA DESCOBERTA SEMÂNTICA

No processo de descoberta semântica, a versão atual do *plugin* faz uso apenas da etapa semântica funcional, do algoritmo do OWL-S *Discovery*. Esta funcionalidade poderia ser estendida para utilizar também a etapa semântica descritiva. Entretanto, isto depende de uma padronização na representação do dicionário de sinônimos necessário para esta etapa.

Outra limitação no que se refere a descoberta semântica é sobre o repositório de serviços. Somente é possível realizar a descoberta em diretórios locais. Este caso também depende que modificações sejam feitas no *OWL-S Discovery*, para que seja permitido efetuar busca em repositórios remotos.

5.6.2 COMPOSIÇÃO AUTOMÁTICA DE SERVIÇOS WEB

O *plugin* OWL-S *Composer* 2.0 realiza a descoberta de composições semanticamente similares, através da comparação de cada serviço. Este processo pode ser considerado semi-automático, visto que o usuário precisa criar a primeira composição manualmente. Outro problema é que, caso não seja encontrado um serviço atômico similar, não é realizada a busca por composições que possam substituir este serviço atômico.

Esta limitação pode ser resolvida utilizando como base o trabalho de (KONA; BANSAL; GUPTA, 2007), onde é apresentado um algoritmo para encontrar composições que atendem aos

requisitos do serviço atômico procurado.

5.6.3 COMPLETUDE

A arquitetura SOA indica três etapas para o ciclo de vida de um serviço Web. No *plugin OWL-S Composer 2.0*, foi incluída apenas a etapa de localização de serviços, não sendo possível efetuar a publicação e execução. Por causa desta limitação, a ferramenta não pode ser considerada completa.

Além disso, são implementados apenas três controladores de fluxo: *Sequence*, *Split* e *Any-Order*. Esta limitação ocorre pois a OWL-S API só implementa estes controladores.

6 VALIDAÇÃO

O *plugin OWL-S Composer 2.0* foi validado de acordo com os requisitos mínimos para uma ferramenta de composição de serviços. Este capítulo apresenta quais são estes requisitos e como o *OWL-S Composer 2.0* atendeu a cada um.

6.1 CRITÉRIOS

Segundo (CHAFLE et al., 2007), os requisitos mínimos que uma ferramenta de composição de serviços deve ter são: Funcionalidade, Interface, Usabilidade e Integração. Além destes, foi proposto por (FONSECA; CLARO; LOPES, 2009) mais um requisito: Legibilidade. E para avaliar a descoberta semântica, este trabalho propõe também o requisito Grau de Similaridade. A seguir é apresentado cada um destes requisitos:

- **Funcionalidade**: o ambiente de desenvolvimento deve suportar todo o ciclo de vida de um serviço Web, desde a criação até a publicação.
- **Interface**: a ferramenta deve oferecer uma interface simples e funcional, de modo que o usuário não necessite utilizar nenhum outro recurso para implementar a composição de serviços.
- **Usabilidade**: a ferramenta deve permitir que o usuário possa acessar todas as funcionalidades de forma intuitiva, através de controles usualmente utilizados, como *menus* e botões.
- **Integração**: os recursos da ferramenta devem ser aproveitados, de forma que todos os elementos que contribuam para a composição de serviços estejam integrados.
- **Legibilidade**: o código OWL-S gerado, referente à composição de serviços, deve ser legível e corresponder sintaticamente e semanticamente a um código válido escrito manualmente.

- **Grau de similaridade**: a ferramenta deve oferecer opções ao usuário quanto ao grau de similaridade semântica que os serviços são encontrados, além de informar o grau correspondente ao apresentar o resultado.

6.2 AVALIAÇÃO DOS CRITÉRIOS NO OWL-S COMPOSER 2.0

O *OWL-S Composer 2.0* atendeu aos critérios descritos na seção 6.1, como é apresentado a seguir:

- **Funcionalidade**: a versão 2.0 do *OWL-S Composer* inclui mais uma etapa do ciclo de vida de um serviço Web, que é a descoberta semântica. A publicação e execução do serviço, no entanto, não são suportadas. Portanto, este requisito está sendo parcialmente atendido, mas poderá ser complementado em trabalhos futuros.

- **Usabilidade**: este requisito é atendido, pois todas as funcionalidades do *plugin* são acessadas através de *menus*, incluindo a descoberta semântica que foi implementada na versão 2.0 como demonstra a figura 17.

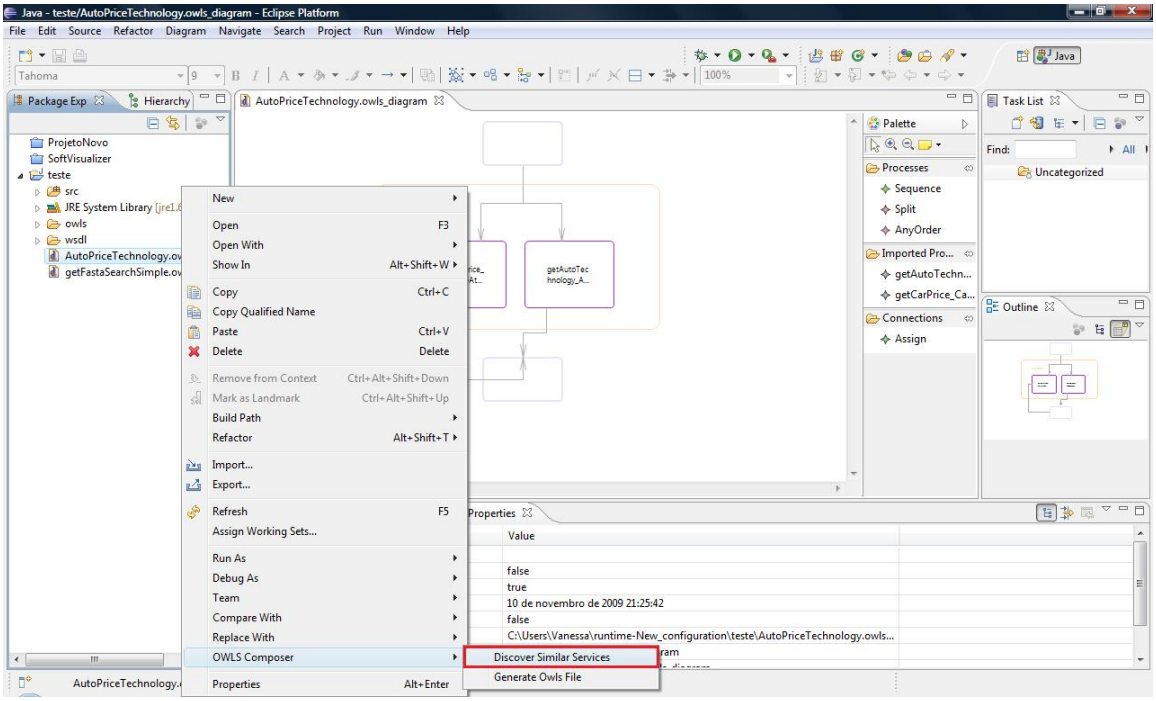

Figura 17: Opção no menu para Descoberta de Serviços no OWL-S Composer 2.0

- **Interface**: este critério é atendido pelo *OWL-S Composer 2.0*, pois este possui uma interface amigável, onde é possível gerar as composições sem precisar de ferramentas extras.

Além disso, a composição similar é gerada automaticamente sem necessitar de interferência do usuário na modelagem do diagrama.

- **Integração**: o *OWL-S Composer 2.0* é um *plugin* para a plataforma Eclipse, a qual já oferece suporte ao desenvolvimento de serviços Web através do WTP. O usuário pode gerar o arquivo WSDL através do WTP e em seguida gerar o OWL-S utilizando o *OWL-S Composer*. Está evidenciada assim, a integração no ambiente de desenvolvimento.

- **Legibilidade**: para verificar este requisito, foi gerado o código OWL-S das composições de serviços e comparados com códigos gerados manualmente. O código apresentou-se legível, de acordo com a especificação da OWL-S 1.1. Segue a descrição do *Profile* de uma das composições geradas pelo *OWL-S Composer 2.0*, no estudo de caso da seção 5.5.

Listagem 6.1: Ontologia Model

```
<!-- Profile description -->
<profile:Profile rdf:ID="SUBSUMESAutoPriceTechnoloyProfile">
   <service:isPresentedBy
        rdf:resource="#SUBSUMESAutoPriceTechnoloyService"/>

   <profile:serviceName xml:lang="en">null</profile:serviceName>

   <profile:hasInput rdf:resource="#Car"/>
   <profile:hasOutput rdf:resource="#RecommendedPrice"/>
   <profile:hasOutput rdf:resource="#Technology"/>
</profile:Profile>
```

- **Grau de similaridade**: o *OWL-S Composer 2.0* apresenta o resultado das composições semanticamente similares de acordo com os graus de similaridade semântica, oferecendo opção ao usuário de filtrar a resposta. Os graus de similaridade disponíveis são *Exact*, *Plugin*, *Subsumes* e *Sibling*. Desta forma, o critério Grau de Similaridade é atendido.

6.3 COMPARAÇÃO COM OUTROS TRABALHOS

Ao comparar o *OWL-S Composer 2.0* com as outras ferramentas estudadas, foram evidenciadas as vantagens deste *plugin*. O CODE (SRINIVASAN; PAOLUCCI; SYCARA, 2005) é uma ferramenta que oferece suporte a todo o ciclo de vida do desenvolvimento de serviços Web semânticos. Porém, a representação visual destes serviços é através de árvores de elementos,

dificultando uma visualização completa e intuitiva. Outra desvantagem é na descoberta dos serviços que é realizada através de análise sintática.

A ferramenta *Discovery and Composition Engine* desenvolvida por (KONA; BANSAL; GUPTA, 2007) realiza descoberta através da análise semântica e a composição é automática. Porém, o resultado não tem opção de serviços similares, somente são retornados os serviços considerados iguais. Outro problema da ferramenta é não estar integrada com um ambiente de desenvolvimento e não apresentar as composições de forma visual.

Diante deste contexto, o *OWL-S Composer 2.0* é uma ferramenta adequada para o desenvolvimento e descoberta de composições de serviços Web semânticos. As vantagens são evidenciadas na utilização interativa de um ambiente gráfico e a implementação dos graus de similaridade semântica para a descoberta.

A tabela 1 apresenta uma comparação entre os trabalhos, através dos critérios apresentados na seção 6.1.

	CODE	Discovery and Composition Engine	OWL-S Composer 2.0
Funcionalidade	Parcial	Parcial	Parcial
Interface	Sim	Sim	Sim
Usabilidade	Não	Não	Sim
Integração	Sim	Não	Sim
Legibilidade	Sim	Sim	Sim
Grau de similaridade	Não	Parcial	Sim

Tabela 1: Tabela comparativa dos trabalhos relacionados

7 CONCLUSÃO

A descoberta automática de serviços Web semânticos é um campo de pesquisa relativamente novo e necessita de mais estudos e ferramentas. A integração da descoberta semântica com as outras etapas do desenvolvimento de serviços Web é importante para os usuários, pois torna a atividade mais fácil e rápida. Contudo, existe uma carência de ferramentas para a plataforma Eclipse que aproveite a potencialidade do *plugin* WTP, e ainda suporte todo o ciclo de vida de um serviço Web.

Este trabalho propôs uma nova versão do *plugin OWL-S Composer*, onde foi incorporada a descoberta semântica através da etapa Semântica Funcional do algoritmo do *OWL-S Discovery*. Desta maneira, o *OWL-S Composer 2.0* oferece suporte a mais uma etapa no ciclo de vida dos serviços Web, a fase da descoberta, além da composição visual já existente. O *OWL-S Composer* versão 2.0, assim como a versão anterior, está disponível no endereço: `https://sourceforge.net/projects/owl-scomposer/`.

7.1 DIFICULDADES ENCONTRADAS

Uma das grandes dificuldades encontradas neste trabalho foi o fato do *plugin OWL-S Composer* possuir pouca documentação, o que tornou difícil o entendimento do código. A ferramenta GMF, utilizada no desenvolvimento do *plugin*, também careceu de documentações, o que dificultou ainda mais o trabalho. O que ajudou durante o desenvolvimento do trabalho foi a lista de discussão do GMF e o contato com o desenvolvedor da primeira versão do *OWL-S Composer*.

7.2 TRABALHOS FUTUROS

Algumas possibilidades de trabalhos futuros já foram indicadas na seção 5.6, estas possibilidades incluem:

- Estender o *OWL-S Composer* para suportar a composição automática, de modo que o usuário não precise criar a primeira composição.
- Estender o *OWL-S Composer* para suportar as etapas de publicação e execução do serviço.
- Implementação de estratégias para detecção, diagnóstico e correção de falhas na execução dos serviços Web através do *plugin*.
- Melhorar o *OWL-S Discovery* para considerar pré-condições e efeitos.
- Trabalhar com os aspectos temporais para a composição de serviços Web.

REFERÊNCIAS

AMORIM, R. Um algoritmo híbrido para descoberta de serviços web semânticos. Trabalho de Conclusão de Curso (Graduação) - Universidade Federal da Bahia. 2009.

BABIK, M. *JAX-SA*. 2008. Último acesso em 27 de outubro de 2009. Disponível em: <http://sourceforge.net/projects/jax-sa/>.

BECHHOFER, S.; HARMELEN, F. van; HENDLER, J.; HORROCKS, I.; MCGUINNESS, D. L.; PATEL-SCHNEIDER, P. F.; STEIN, L. A. *OWL Web Ontology Language Reference*. Fevereiro 2004. Último acesso em 18 de outubro de 2009. Disponível em: <http://www.w3.org/TR/owl-ref/>.

BERNERS-LEE, T.; HENDLER, J.; LASSILA, O. The semantic web. *Scientific American*, Maio 2001.

CHAFLE, G.; DAS, G.; DASGUPTA, K.; KUMAR, A.; MITTAL, S.; MUKHERJEA, S.; SRIVASTAVA, B. An integrated development enviroment for web service composition. *IEEE International Conference on Web Services*, p. 839–847, 2007.

CLARO, D. B. Reactive planning to discover and compose web services. *Doctoral Consortium of Enterprise Information Systems*, 2004.

CLARO, D. B.; ALBERS, P.; HAO, J.-K. Approaches of web services composition. *International Conference on Enterprise Information Systems*, p. 208–213, Maio 2005.

CLARO, D. B.; MACêDO, R. J. de A. Serviços web e sua relação com sistemas de informação. *IV Simpósio Brasileiro de Sistemas de Informação*, 2008.

CMU. *OWL-S Intergrated Development Environment*. Julho 2005. Último acesso em 25 de junho de 2009. Disponível em: <http://projects.semwebcentral.org/projects/owl-s-ide>.

EMF. *Eclipse Modeling Framework Project*. 2009. Último acesso em 22 de agosto de 2009. Disponível em: <http://www.eclipse.org/modeling/emf/>.

FONSECA, A. de A.; CLARO, D. B.; LOPES, D. C. P. Gerenciando o desenvolvimento de uma composição de serviços web semânticos através do owl-s composer. *V Simpósio Brasileiro de Sistema de Informação*, p. 109–120, 2009.

GEF. *Graphical Editing Framework*. 2009. Último acesso em 23 de agosto de 2009. Disponível em: <http://www.eclipse.org/gef/>.

GMF. *Graphical Modeling Framework*. 2009. Último acesso em 22 de outubro de 2009. Disponível em: <http://www.eclipse.org/modeling/gmf/>.

JET. *Java Emitter Templates*. 2009. Último acesso em 26 de junho de 2009. Disponível em: <http://www.eclipse.org/modeling/m2t/?project=jet>.

KLUSCH, M.; FRIES, B.; SYCARA, K. Automated semantic web service discovery with owls-mx. *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, p. 915–922, 2006.

KONA, S.; BANSAL, A.; GUPTA, G. Automatic composition of semantic web services. *2007 IEEE International Conference on web Services*, p. 150–158, 2007.

KOPECKY, J.; VITVAR, T.; BOURNEZ, C.; FARRELL, J. Sawsdl: Semantic annotations for wsdl and xml schema. *IEEE Internet Computing*, v. 11, n. 6, p. 60–67, 2007.

KRUG, A.; KüSTER, U.; RUHMER, C. *OPOSSum Online Portal for Semantic Services*. Agosto 2009. Último acesso em 10 de novembro de 2009. Disponível em: <http://fusion.cs.uni-jena.de/opossum/index.php>.

LAUSEN, H.; POLLERES, A.; ROMAN, D. Web service modeling ontology (wsmo). *W3C Member Submission*, Junho 2005. Último acesso em 07 de novembro de 2009. Disponível em: <http://www.w3.org/Submission/WSMO/>.

LOPES, D. C. P.; HAMMOUDI, S.; SOUZA, J. de; BONTEMPO, A. Metamodel matching: Experiments and comparison. *International Conference on Software Engineering Advances*, p. 2, 2006.

MARTIN, D.; BURSTEIN, M.; HOBBS, J.; LASSILA, O.; MCDERMOTT, D.; MCILRAITH, S.; NARAYANAN, S.; PAOLUCCI, M.; PARSIA, B.; PAYNE, T.; SIRIN, E.; SRINIVASAN, N.; SYCARA, K. Owl-s semantic markup for web services. *Internet Computing, IEEE*, p. 72–81, 2004.

MICROSOFT. *DCOM: Distributed Component Object Model*. 1995. Último acesso em 16 de novembro de 2009. Disponível em: <http://www.microsoft.com/com/default.mspx>.

MINDSWAP. *Maryland Information and Network Dynamics Lab Semantic Web Agents Project*. 2007. Último acesso em 13 de agosto de 2009. Disponível em: <http://www.mindswap.org/2004/owl-s/services.shtml>.

NEWCOMER, E. *Understanding Web Services: XML, WSDL, SOAP and UDDI*. [S.l.]: Addison-Wesley Longman Publishing, 2002.

OMG. *CORBA: Common Object Request Broker Architecture*. 1995. Último acesso em 16 de novembro de 2009. Disponível em: <http://www.corba.org/>.

PAOLUCCI, M.; KAWAMURA, T.; PAYNE, T. R.; SYCARA, K. Semantic matching of web services capabilities. *Lecture Notes In Computer Science, Springer*, p. 333–347, 2002.

POTTS, S.; KOPACK, M. *Teach Yourself Web Services in 24 hours*. [S.l.]: Sams, 2003.

SAMPER, J. J.; ADELL, F. J.; BERG, L. van den; MARTINEZ, J. J. Improving semantic web service discovery. *Journal of networks*, v. 3, n. 1, 2008.

SPERBERG-MCQUEEN, C. M.; THOMPSON, H. *XML Schema*. Abril 2000. Último acesso em 27 de outubro de 2009. Disponível em: <http://www.w3.org/XML/Schema>.

SRINIVASAN, N.; PAOLUCCI, M.; SYCARA, K. *CODE: A Development Environment for OWL-S Web Services*. [S.l.], Outubro 2005.

WTP. *Web Tools Platform (WTP) Project*. 2009. Último acesso em 04 de setembro de 2009. Disponível em: <http://www.eclipse.org/webtools/>.