UNIFACS UNIVERSIDADE SALVADOR
PROGRAMA DE PÓS-GRADUAÇÃO
MESTRADO PROFISSIONAL EM SISTEMAS E COMPUTAÇÃO

CLEBER JORGE LIRA DE SANTANA

SWS-EDITOR: UMA FERRAMENTA BASEADA EM SERVIÇOS PARA ANOTAÇÃO SEMÂNTICA DE SERVIÇOS WEB *RESTFUL*

Salvador
2015

CLEBER JORGE LIRA DE SANTANA

SWS-EDITOR: UMA FERRAMENTA BASEADA EM SERVIÇOS PARA ANOTAÇÃO SEMÂNTICA DE SERVIÇOS WEB *RESTFUL*

Dissertação apresentada ao Mestrado em Sistemas e Computação da UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial para a obtenção do título de Mestre.

Orientador: Prof. Dr. Paulo Caetano da Silva.

Salvador
2015

FICHA CATALOGRÁFICA

(Elaborada pelo Sistema de Bibliotecas da UNIFACS Universidade Salvador, Laureate International Universities)

Lira de Santana, Cleber Jorge

SWS – Editor: uma ferramenta baseada em serviços para anotação semântica de serviços Web *Restful* / Cleber Jorge Lira de Santana. - 2015.

171 f. : il.

Dissertação Programa de Pós-Graduação em Sistemas e Computação de UNIFACS Universidade Salvador, Laureate International Universities como requisito parcial à obtenção do título de Mestre em Sistemas e Computação.

Orientador: Prof. Dr. Paulo Caetano da Silva.

1. Engenharia de software. 2. Web semântica. 3. Ontologia. I. da Silva, Paulo Caetano, orient. II. Título.

CDD: 005. 1

CLEBER JORGE LIRA DE SANTANA

SWS-EDITOR: UMA FERRAMENTA BASEADA EM SERVIÇOS PARA ANOTAÇÃO SEMÂNTICA DE SERVIÇOS WEB *RESTFUL*

Dissertação aprovada como requisito parcial para obtenção do grau de Mestre em Sistemas e Computação, na UNIFACS Universidade Salvador, Laureate International Universities, pela seguinte banca examinadora:

Paulo Caetano da Silva – Orientador _____
Ph.D. in Computer Science, Universidade Federal de Pernambuco - UFPE
UNIFACS Universidade Salvador, Laureate International Universities

Cássio Vinícius Serafim Prazeres _____
Doutor Em Ciências da Computação e Matemática Computacional pela Universidade de São Paulo - USP/ICMC
Universidade Federal da Bahia - UFBA

Artur Henrique Kronbauer _____
Doutor em Ciência da Computaçãopela Universidade Federal da Bahia - UFBA
UNIFACS Universidade Salvador, Laureate International Universities

Salvador, 21 de setembro de 2015.

Dedico este trabalho aos meus pais pelo Amor, Respeito, Sabedoria e Perseverança ensinados ao longo da minha caminhada.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por nunca ter me abandonado e me deixado desistir.

A minha mãe e meu pai por batalhar e acreditar no investimento da minha educação.

Aos meus antigos colegas de trabalho da SEFAZ que contribuíram com discussões, ideias, apoio técnico e diversas formas de incentivo.

Ao Professor Paulo Caetano da Silva, pela contribuição na definição do objeto pesquisado e pela orientação, ajuda, análise, crítica, dedicação, paciência e pelo crédito depositado.

Aos demais professores da UNIFACS pelas esclarecedoras aulas e competência na arte de ensinar e mostrar os caminhos que levam ao conhecimento.

À minha família: minhas irmãs Carine, Kelly, Caliane por compreenderem a minha inquietação e pelo apoio nos momentos mais difíceis da minha vida; meu cunhado Adailton pelo espírito colaborador; minha sobrinha Ana Clara pela alegria sempre contagiante trazendo motivação nos momentos de fraqueza; minha avó por fornecer apoio na minha educação.

Ao senhor Deobaldo (*in memorian*) por ter colaborado para o desenvolvimento da minha educação nos primórdios da graduação através do qual possibilitou inclusive está concluindo esse Mestrado.

A minha esposa, que tem sido um alicerce na minha caminhada pessoal e profissional fornecendo amor e companheirismo.

Ao IFBA que proporcionou o ambiente propício e o apoio ao trabalho desenvolvido.

Valeu a pena? Tudo vale a pena
Se a alma não é pequena.
Quem quer passar além do Bojador
Tem que passar além da dor.
Deus ao mar o perigo e o abismo deu, Mas nele é que espelhou o céu.

Fernando Pessoa

Meus filhos terão computadores, sim, mas antes terão livros. Sem livros, sem leitura, os nossos filhos serão incapazes de escrever inclusive a sua própria história.

Bill Gates

RESUMO

Os Serviços Web possibilitaram que as organizações integrassem suas aplicações de *softwares*, viabilizando a automação de processos de negócios em diversas áreas. Entretanto, é necessário fornecer soluções que favoreçam a seleção, descoberta e integração desses Serviços Web em tempo de execução de maneira eficaz e eficiente. Para favorecer a automação nas tarefas de requisição, descoberta e composição de Serviços Web algumas ferramentas propõem a realização da anotação semântica, entretanto, divergem com relação às funcionalidades oferecidas e a adequação às tecnologias que são recomendações do W3C, além do uso ser restrito a ambientes específicos (i.e. utilização apenas em ambiente web ou *desktop*). A utilização de serviços para realizar a anotação semântica de Serviços Web favorece ao reuso da solução em diferentes ambientes (e.g. *desktop*, web, *mobile*) possibilitando dessa forma alcançar um maior número de usuários. Na arquitetura da web atual muitos dos Serviços Web são projetados conforme o estilo arquitetural REST**,** também conhecidos como Serviços Web *restful* baseados no protocolo HTTP. Este trabalho apresenta uma ferramenta baseada em serviços, denominada SWS Editor (*Semantic Web Services Editor*) que faz uso de ontologias OWL a partir da especificação SAWSDL para realizar a anotação semântica de Serviços Web *restful* descritos sintaticamente em WSDL. A arquitetura da ferramenta proposta possibilitará a criação de um repositório de Serviços Web Semânticos que pode ser utilizado para recuperação automática de serviços. Como prova de conceito, é discutido um exemplo de uso da ferramenta através do qual são apresentadas as suas funcionalidades, consumo dos serviços e a recuperação semântica através de consultas SPARQL.

Palavras Chaves: *Web Service restful*. Anotação semântica. SAWSDL. Ontologia. SWS Editor.

ABSTRACT

Web services have enabled organizations to integrate their software applications, enabling the automation of business processes in several areas. However, it is necessary to provide solutions that favor the selection, discovery and integration of Web services at runtime effectively and efficiently. To promote automation in the request, discovery and composition tasks of Web services, some tools propose the realization of semantic annotation; however, they differ with respect to the features offered and the adequacy to the technologies that are W3C recommendations, and the use is restricted to specific environments (i.e. use only on web environment or desktop). The use of services to perform the semantic annotation of Web Services encourages the reuse of the solution in different environments (e.g. desktop, web, mobile) thus enabling to reach a higher number of users. In the current web architecture, many Web services are designed according to the REST architectural style, also known as Restful Web Services based on the HTTP protocol. This paper presents a service-based tool, called SWS Editor (Semantic Web Services Editor) that makes use of OWL ontologies from SAWSDL specification to perform the semantic annotation of Restful Web Services syntactically described in WSDL. The architecture of the proposed tool will enable the creation of a Semantic Web Services repository that can be used for automatic recovery services. As proof of concept, an example of use of the tool is discussed, where its features, service consuming and semantic retrieval via SPARQL queries are presented.

Keywords: Web Service restful. Semantic annotation. SAWSDL. Ontology. SWS Editor.

LISTA DE ILUSTRAÇÕES

LISTA DE QUADROS

LISTAGEM DE CÓDIGO

LISTA DE ABREVIATURAS E SIGLAS

AJAX	Asynchronouus Java Script and XML
API	Application Programming Interface
BEEP	Blocks Extensible Exchange Protocol
BLL	Business Logic Layer
CORBA	Common Object Request Broker Architecture
DAL	Data Access Layer
DCOM	Distributed Component Object Model
DML	Data Manipulation Language
DOM	Document Object Model
EbXML	eXtensible Markup Language
FTP	File Transfer Protocol
hRests	HTML for RESTfull Services
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IDE	Integrated Development Environment
IFBA	Instituto Federal de Educação Ciência e Tecnologia da Bahia
IDL	Interface Definition Language
IIOP	Internet Inter - ORB Protocol
IIS	Internet Information Service
IRI	Internationalized Resource Identifier
JRMP	Java Remote Method Protocol
JSON	JavaScript Object Notation

MSA	Mechanism for semantic annotation
MVC	Model View Controller
N3	Notation3
NASSL	Network Application Service Specification Language
OMG	Object Management Group
ORPC	Object Remote Procedure CALL
OWL	Web Ontology Language
PDF	Portable Document Format
QL	Query Language
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
REST	Representational State Transfer
RL	Rule Language
RMI	Remote Method Invocation
ROA	Resource-Oriented Architecture
RPC	Remote Procedure Call
SA-REST	Semantic Annotation of Web Resources
SAWSDL	Semantic Annotations for WSDL and XML Schema
SDL	Service Description Language
SGML	Standard Generalized Markup Language
SMTP	Simple Mail Transfer Protocol
SOA	Service-Oriented Architecture
SOAP	Simple Object Acess Protocol
SPARQL	SPARQL Protocol and RDF Query Language
SWEET	Semantic Web Services Editing Tool
TCP/IP	Transmission Control Protocol/Internet Protocol

Turtle	Terse RDF Triple Language
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
URI	Uniform resource identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
WADL	Web Application Description Language
WSDL	Web Services Descriptions Language
WSDL-S	Web Service Semantics
WSMO	Web Service Modeling Ontology
WSMO Studio	Web Service Modelling Ontology Studio
WSMX3	Web Service Modelling eXecution environment
WWW	World Wide Web
XML	eXtensible Markup Language
XSD	XML Schema Definition
XSL	eXtensible Stylesheet Language
XSLT	eXtensible Stylesheet Language for Transformation

SUMÁRIO

1 INTRODUÇÃO

Este capítulo tem como propósito contextualizar as questões relacionadas a anotação semântica de Serviços Web *restful* que proporcionam a automação das tarefas de solicitar, descobrir e compor esses serviços, em seguida justificar porque essas questões necessitam ser analisadas e por último descrever a motivação para a realização desta dissertação, juntamente com os objetivos que se pretendem alcançar. Sua organização ocorre da seguinte forma: a Seção 1.1 contextualiza os problemas que serão discutidos neste trabalho; a Seção 1.2 justifica porque esses problemas devem ser resolvidos e para os quais serão apresentados a proposta de solução; a Seção 1.3 descreve os aspectos que motivaram a realização desta dissertação; a Seção 1.4 detalha os objetivos; a Seção 1.5 discorre sobre a metodologia de pesquisa e finalmente, a Seção 1.6 apresenta a estrutura organizacional deste trabalho.

1.1 CONTEXTUALIZAÇÃO

No início da década de 80, Tim Berners-Lee[1] começou a desenvolver um sistema de navegação hipertexto com o principal objetivo de facilitar o registro de documentos em um repositório e que facilitasse a apresentação da informação. Esse projeto, denominado WWW (*World Wide Web*), unia as técnicas de recuperação da informação e de hipertexto para construir um sistema de informação de fácil uso e de acesso global. Tim Berners-Lee publicou a proposta[2] da WWW através do qual afirma que o seu objetivo é permitir que o acesso à informação pudesse ser realizado de qualquer lugar.

Esse projeto evoluiu e no início da década de 90 a web estava disponível. Inicialmente a Internet foi utilizada para publicação de documentos científicos. De acordo com Bizer (2009), a Web inicial foi denominada de Web dos documentos por ser um espaço global através do qual a informação possui como ponto de conexão os *hyperlinks* permitindo assim a navegação entre os documentos.

Entretanto, além do que foi proposto inicialmente, a web evoluiu desde a década de 90 e passou a se tornar um imenso repositório de serviços necessários às organizações para execução de atividades e em alguns cenários auxiliam na tomada de decisões nas mais diversas áreas. Esses serviços, que estão disponibilizados na Internet, essenciais às

[1] http://www.w3.org/People/Berners-Lee/

[2] http://www.w3.org/Proposal

organizações públicas e privadas são conhecidos como Serviços Web[3]. Essa evolução da web marcada pelo desenvolvimento de aplicações interativas, possibilitando um maior envolvimento dos usuários e que enfatiza a comunicação e a colaboração é denominada Web 2.0.

A Web 2.0, também conhecida como Web Social, refere-se à segunda geração da web, na qual aplicações e Serviços Web são interoperáveis e centrados no usuário que promovem a conectividade social, o compartilhamento de mídia e informações, a geração de conteúdo criado pelo usuário e a colaboração entre indivíduos e organizações. De acordo com O' Relly (2005), a Web 2.0 é a Internet como plataforma envolvendo aplicações baseadas em *folksonomia*[4], *wikis*, redes sociais e abrange todos os dispositivos conectados. Por essa razão, nessa segunda geração da web, a publicação e o acesso à informação tornaram-se ações de fácil execução para quaisquer indivíduos. Nessa segunda geração da web o conteúdo exposto pelas aplicações é em grande parte gerada pelos seus usuários o que facilita o acesso e reuso desse conteúdo por outras aplicações, pois suas interfaces estão disponíveis como Serviços Web. Com o desenvolvimento dos Serviços Web, tornou possível requisitar, publicar e localizar aplicações na Internet. *Web Service* é uma aplicação de *software* que consegue receber solicitações de outras aplicações de *software* e, posteriormente, gerar as respostas que possam ser interpretadas por elas. Uma das principais características desses serviços é a interoperabilidade que está relacionada com a capacidade de uma aplicação se comunicar com outra aplicação independentemente da plataforma de construção e execução. Entretanto, os Serviços Web, na forma como são construídos atualmente não possibilitam a automação na comunicação entre essas aplicações.

Um *Web Service* quando disponibilizado na web possui apenas sua descrição sintática, o que inviabiliza que qualquer tipo de informação possa ser interpretado por um sistema computacional. Apenas a descrição sintática fornecida pelos Serviços Web não permite estabelecer uma automação na utilização desses serviços. Qualquer serviço que necessite fazer uso de outro precisa passar por adaptações que serão realizadas por desenvolvedores de *software* para conseguir utilizá-lo. Portanto, é necessário estabelecer mecanismos para que Serviços Web possam, por exemplo, serem localizados e compostos de forma automatizada e que atenda as solicitações do usuário final. Em casos especiais, se nenhum *Web Service*

[3] http://www.w3schools.com/webservices/

[4] http://vanderwal.net/folksonomy.html

satisfizer a funcionalidade solicitada pelo usuário, deverá haver a possibilidade de combinar Serviços Web existentes a fim de atender a requisição. Para atender a essa necessidade, em paralelo ao desenvolvimento da segunda geração da web, um novo modelo da web passa a ser discutido. É a terceira geração da web, denominada Web Semântica[5]. A Web Semântica propõe que componentes de *software* possam processar automaticamente solicitações originadas do usuário, sendo capaz de analisar todo o contexto de uma determinada requisição sem a necessidade da intervenção, por exemplo, de um desenvolvedor de *software*.

Foi realizada uma revisão bibliográfica inicialmente com o objetivo de investigar o universo de aplicações na área da Web Semântica, mais especificamente os Serviços Web. A partir dessa revisão observou-se que grande parte das pesquisas acerca de Serviços Web semânticos está relacionada a arquitetura SOAP, sendo reduzido o número de pesquisas na arquitetura REST. Nessa revisão da literatura foram considerados os questionamentos a seguir:

a) Quais são as ferramentas para a implementação de Serviços Web semânticos?
b) Quais tecnologias são consideradas para a implementação de Serviços Web semânticos construídos conforme a arquitetura REST?
c) Existe um padrão determinado por alguma instituição para a descrição sintática de Serviços Web construídos conforme o estilo arquitetural REST?
d) É possível realizar a anotação semântica de Serviços Web construídos conforme a arquitetura REST?
e) A utilização de serviços torna a arquitetura da solução extensível?

Conclui-se que soluções baseadas em serviços para a anotação semântica em Serviços Web implementados de acordo com a arquitetura REST é um desafio a ser enfrentado pelos pesquisadores dessa área da ciência da computação, pois poucas foram as soluções encontradas.

1.2 JUSTIFICATIVA

De acordo com Oliveira (2009), Serviços Web Semânticos potencializa os processos de recuperação, extração e manutenção da informação viabilizando a Web Semântica e tornando possível às aplicações de *software* requisitar e executar serviços sem a necessidade da intervenção de um desenvolvedor de *software*. Com a Web 2.0, muitos Serviços Web são construídos conforme o estilo arquitetural REST, entretanto, esses serviços fornecem apenas a

[5] http://www.w3.org/standards/semanticweb/

sua descrição sintática e por essa razão necessitam da anotação semântica, motivo de elaboração deste trabalho, que proporcionarão a automação em processos de integração entre aplicações de *software*, maior eficiência na realização de buscas na web, entre outros.

A escassez de ferramentas para realizar a anotação semântica de Serviços Web *restful* e que não seguem padrões como os recomendados pelo W3C bem como fornecer uma solução baseada em serviços, justifica a elaboração deste trabalho. Muito se discute na indústria de *software* a respeito sobre a necessidade da descrição sintática de Serviços Web construídos conforme o padrão REST. Desde então surgiram alguns padrões para realizar tais descrições: A WADL define arquivos baseados em XML para realizar a descrição sintática de aplicações baseadas na WEB, porém, não se tornou um padrão de fato. Por não existir um consenso com relação a descrição sintática dos Serviços Web *restful* motivado pela utilização do protocolo HTTP para realizar as operações através dos seus métodos, e.g. GET, POST, esta dissertação optou por utilizar a linguagem WSDL por se tratar de uma recomendação W3C e que por esse motivo possivelmente poderá ser adotado no futuro por desenvolvedores de *software*. A WSDL é um padrão especificado pelo W3C que define arquivos XML para descrição sintática de Serviços Web e na sua versão 2.0 fornece o suporte necessário para descrição sintática de Serviços Web *restful*. Com a anotação semântica sendo realizada em documentos WSDL, os mesmos terão a capacidade de descrever as funcionalidades sintáticas bem como semânticas dos Serviços Web.

Foi identificado na pesquisa bibliográfica, propostas semelhantes em relação à discussão desenvolvida neste trabalho, entretanto, as soluções encontradas divergem com relação aos requisitos implementados. Por essa razão, essas divergências levam a necessidade de estabelecer um conjunto de requisitos para a implementação de uma ferramenta que se propõe a anotar semanticamente Serviços Web *restful* baseados na linguagem WSDL.

1.3 MOTIVAÇÃO

De acordo com (BUDINOSKI et al, 2010), Serviços Web construídos utilizando a arquitetura .NET [6] e o *framework* J2EE [7] não possuem semântica devido as limitações existentes na arquitetura da web atual e por conflitos (i.e. não suportam a interoperabilidade) que podem existir em padrões de tecnologias proprietárias, o que inviabiliza realizar automaticamente a descoberta de serviços.

[6] https://msdn.microsoft.com/en-us/library/vstudio/a4t23ktk(v=vs.100).aspx
[7] http://www.oracle.com/technetwork/java/javaee/overview/index.html

Guttula (2012) afirma que o número de ferramentas e aplicações disponíveis como Serviços Web têm aumentado drasticamente nos últimos anos. Por exemplo, de maio de 2010 a abril de 2012, a *BioCatalogue*[8], um repositório de serviços de ciências biológicas, aumentou o número de Serviços Web biomédicos de 1627 para 2290. De maneira similar, Seekda[9], um repositório serviços de propósito geral, descobriu mais de 28.000 Serviços Web em um período de 68 meses. Esses aumentos representam um desafio para usuários que desejam encontrar o *Web Service* certo para realizar um determinado trabalho.

De acordo com Viana (2012), dentro do contexto SOA [10] (*Service-Oriented Architecture)* e com a tendência crescente do uso de redes sem fio e 3G, as organizações vêm disponibilizando em larga escala os Serviços Web com o intuito de atender as solicitações dos usuários de forma rápida em diversas situações. Em diversos cenários, existe a necessidade de que dois ou mais serviços de diferentes fabricantes sejam automaticamente compostos a fim de atender a uma solução única. Portanto, faz-se necessária uma solução de anotação semântica para tornar autônoma as tarefas de execução, descoberta, invocação e composição de Serviços Web. De acordo com (OREN et al., 2006), a anotação semântica é o processo que possibilita a inserção de dados em algum outro conjunto de dados em uma série de domínios distintos (e.g.: anotação de documentos, *wikis*, *blogs*, *tagging*) podendo ser conduzida de maneira manual, semiautomática ou completamente automática permitindo que *softwares* possam interpretar, combinar e utilizar esses dados na web.

Grande parte dos Serviços Web expostos na web estão implementados seguindo o padrão SOAP (*Simple Object Acess Protocol*)[11] que é baseado em RPC (*Remote Procedure Call*)[12], entretanto, os serviços projetados para REST (*Representational State Transfer*)[13] que é definido sob o protocolo HTTP[14] passam a se proliferar na web e o seu uso se tornou muito difundido na Web 2.0. Por essa razão se faz necessário também fazer com que os serviços escritos conforme o paradigma REST sejam interpretáveis por sistemas computacionais, ou seja, existe a necessidade de anotar semanticamente esses Serviços Web utilizando

[8] Web Service Registry For The Life Science Community," in Nature Precedings, 2009

[9] http://www.seekda.com/

[10] http://serviceorientation.com/whatissoa/fundamental_design_terminology_and_concepts

[11] http://www.w3.org/TR/soap/

[12] http://web.cs.wpi.edu/~cs4514/b98/week8-rpc/wee8-rpc.html

[13] http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm

[14] http://www.w3.org/Protocols/Activity.html

tecnologias que sejam especificações do W3C (*World Wide Web Consortium*)[15]. REST é um estilo arquitetural para sistemas hipermídias distribuídos que enfatiza a generalização das interfaces, a escalabilidade da interação entre os componentes e a instalação dos mesmos (FIELDING, 2000). De acordo com Richardson e Ruby (2007), os Serviços Web implementados com base em uma arquitetura REST são denominados de Serviços Web *restful.*

A anotação semântica de Serviços Web *restful* proporciona novos desafios visto que as soluções existentes para anotar semanticamente esses serviços utilizam diferentes especificações diante de um cenário no qual são encontradas descrições sintáticas dos Serviços Web realizadas em páginas HTML, documentos WADL (*Web Application Description Language*)[16], documentos WSDL (Serviços Web *Descriptions Language*)[17]. Por esse motivo, as pesquisas na área de mecanismos de anotação semântica, possibilitaram a formação de algumas especificações como SAWSDL (*Semantic Annotations for WSDL and XML Schema*)[18], WSDL-S (*Web Service Semantics*)[19] e SA-REST (*Semantic Annotation of Web Resources*)[20]. Essas especificações são utilizadas por aplicações de *software* que possuem como principal objetivo fornecer semântica a serviços e aplicações web construídas sob o padrão REST e/ou SOAP. A especificação SAWSDL é uma das mais recentes recomendações do W3C e suporta a realização das anotações semânticas diretamente em documentos WSDL através de conceitos descritos em determinada linguagem semântica identificados por URIs[21] e que, consequentemente, fornece uma estrutura que elimina adaptações durante o seu uso (PRAZERES, 2009).

De acordo com Segev e Sheng (2009), as ontologias são utilizadas por diversas aplicações da Web Semântica, pois possuem a capacidade de modelar conceitos e seus relacionamentos em um determinado domínio. Por isso, segundo (PATIL et al., 2004), existem ferramentas e *frameworks* que se propõe fornecer a anotação semântica dos Serviços Web utilizando ontologias.

---

[15] http://www.w3.org/Consortium/

[16] https://wadl.java.net/

[17] http://www.w3.org/TR/wsdl20/

[18] http://www.w3.org/TR/sawsdl/

[19] http://www.w3.org/Submission/WSDL-S/

[20] http://www.w3.org/Submission/SA-REST/

[21] http://www.w3.org/wiki/URI

Aplicações de *software* da Web 2.0 expõe suas funcionalidades como Serviços Web *restful* (MALESHKOVA; PEDRINACI; DOMINGUE, 2009), entretanto, os autores afirmam que não há um padrão amplamente aceito para descrever sintaticamente os Serviços Web *restful*. Por essa razão, a linguagem WSDL 2.0 foi projetada para descrever sintaticamente esses serviços sendo uma recomendação do W3C desde junho de 2007 (MANDEL, 2008).

Portanto, é importante a incorporação da anotação semântica, a partir do uso de ontologias, em Serviços Web *restful* descritos sintaticamente em documentos WSDL. Este trabalho fornece uma solução baseada em serviços que utiliza ontologia para auxiliar os desenvolvedores de *software* na tarefa de anotação semântica. Para Erl (2009), o desenvolvimento baseado em serviços beneficia os reuso das aplicações de *software* pois possibilita que estas aplicações possam ser utilizadas em ambientes distintos.

1.4 OBJETIVOS

O objetivo deste trabalho é apresentar uma solução baseada em serviços que possibilite a anotação semântica de Serviços Web *restful* descritos em linguagem WSDL. Para atingir o objetivo deste trabalho é necessário alcançar os seguintes objetivos específicos:

a) Realizar um estudo sobre a Web Semântica no que se refere à arquitetura, principais tecnologias e aplicações dessa nova fase da web. O objetivo da realização desse estudo é entender os conceitos e tecnologias (e.g.: ontologias, OWL, RDF, SPARQL, entre outros) que formam a base da Web Semântica para que a solução proposta possa ser adotada por um número maior de desenvolvedores de *software*;
b) Apresentar um estudo sobre o desenvolvimento de serviços e Serviços Web apontando as arquiteturas existentes para sua implementação, determinando suas principais diferenças e identificar vantagens e desvantagens. O entendimento na implementação de serviços e Serviços Web possibilitará a construção de uma ferramenta que possa ser acessada a partir de diferentes ambientes (e.g.: *desktop*, web, *mobile*);
c) Discutir os Serviços Web semânticos *restful*, apresentando as principais tecnologias implementadoras com o intuito de utilizá-las na elaboração deste trabalho;
d) Desenvolver uma ferramenta baseada em serviços para a anotação semântica de Serviços Web *restful* com base nas decisões tomadas a partir dos estudos anteriores;
e) Realizar um exemplo de uso, baseado em uma solução *desktop*, como prova de conceito para testar a solução proposta nesta dissertação.

Ao final da elaboração deste trabalho, todos os artefatos construídos serão disponilizados em um repositório na internet.

1.5 METODOLOGIA DE PESQUISA

Barros (2000), afirma que a pesquisa aplicada possui o objetivo de contribuir para fins práticos, visando a solução imediata ou não do problema encontrado na realidade. Este trabalho representa uma pesquisa aplicada através do qual seu resultado é demonstrado através de exemplo de uso (Capítulo 5), com a busca detalhada do conhecimento e tecnologias necessárias para obter maior familiaridade ao problema e a solução pretendida. A classificação da pesquisa desenvolvida neste trabalho está ilustrada no Quadro 1-1.

Quadro 1-1 Quadro metodológico

Método de procedimento	**Pesquisa Bibliográfica**
Variáveis	Independentes: Boas práticas, ferramentas, serviços Dependente: Facilitar a anotação semântica Serviços Web *restful* em linguagem WSDL

O método de procedimento definido para a pesquisa é:

(i) Pesquisa Bibliográfica: baseado em referências bibliográficas. Essas referências são coletadas a partir de fontes escritas podendo ser: documentos, que são obras escritas, artigos científicos, livros, entre outros.

A natureza das variáveis da pesquisa é qualitativa. Para Marconi e Lakatos (2007), o paradigma qualitativo preocupa-se em analisar e descrever a complexidade do comportamento humano, fornecendo análises mais detalhadas sobre as investigações, hábitos, atitudes, tendências de comportamento, etc. Os métodos qualitativos proveem informações mais exploratórias.

As principais etapas que constituíram o ciclo dessa pesquisa são apresentadas no Quadro 1-2.

Quadro 1-2 Etapas da Pesquisa

➢ **Pesquisa bibliográfica:**
Nesta etapa é realizada pesquisa sobre a base teórica que envolve esse trabalho. Foi feito um estudo sobre Web Semântica e Serviços Web além de identificar os trabalhos correlatos.
➢ **Estudo sobre Serviços e Serviços Web:**
Nessa fase são explorados os conceitos que norteiam os Serviços e Serviços Web. Padrões já utilizados pela indústria de *software* são apresentados bem como padrões mais recentes que são a base para o desenvolvimento de APIs Web.
➢ **Estudo sobre Serviços Web semânticos:**
Já nesta etapa o estudo envolveu o aprendizado sobre anotações semânticas de Serviços Web e tópicos específicos dessa área.
➢ **Leitura sobre trabalhos correlatos:**
Nessa fase são analisados os trabalhos identificados na primeira etapa. Foram levantados os trabalhos com objetivos similares ao proposto nesta dissertação.
➢ **Projeto e Desenvolvimento:**
Nessa etapa é projetada e desenvolvida uma ferramenta para anotação semântica dos Serviços Web *restful.*
➢ **Exemplo de Uso:**
Nessa etapa é realizada a anotação semântica de Serviços Web *restful* utilizando a ferramenta desenvolvida e o objetivo principal é executar testes de "caixa preta" com o intuito de demonstrar a operacionalidade da ferramenta desenvolvida.

Com base nesse enfoque, foi construída uma ferramenta baseada em serviços para realizar a anotação semântica de Serviços Web *restful* descritos sintaticamente em linguagem WSDL submetido a um exemplo de uso, utilizado como prova de conceito, no qual foram levantadas as questões referentes ao seu uso, agregando críticas e sugestões de melhoria.

1.6 ESTRUTURA DA DISSERTAÇÃO

Esse trabalho está dividido em seis capítulos. O Capítulo 1 trata-se da introdução e contextualiza este trabalho, apresentando seus objetivos, etapas realizadas durante o desenvolvimento desta pesquisa, motivação e justificativa e os objetivos a serem alcançados.

O Capítulo 2 discute as tecnologias envolvidas que subsidiaram a elaboração desta proposta.

O Capítulo 3 apresenta a revisão bibliográfica. São discutidos os trabalhos correlatos desenvolvidos relacionados com as questões de pesquisa desta dissertação. Esses trabalhos são comparados em uma discussão, levando-se em conta as vantagens e desvantagens de cada um e que servirão como base para a construção da ferramenta proposta nesse trabalho.

No Capítulo 4 é apresentada a ferramenta proposta e discute-se a sua concepção, arquitetura e implementação. São demonstradas as técnicas utilizadas para construção da ferramenta.

O Capítulo 5 refere-se a um exemplo de utilização da ferramenta como prova de conceito.

O Capítulo 6 discute as considerações finais sobre o trabalho, uma descrição das principais contribuições para área da computação, publicações e sugestões para trabalhos futuros.

2 TECNOLOGIAS PARA ANOTAÇÃO SEMÂNTICA DE SERVIÇOS WEB

Algumas tecnologias, discutidas a seguir, são utilizadas no desenvolvimento da ferramenta proposta: XML, *XML SCHEMA DEFINITION*, WSDL, SERVIÇOS WEB, SAWSDL, RDF, OWL, SPARQL. Nas seções a seguir são descritos os principais conceitos dessas tecnologias.

2.1 XML

A XML (*eXtensible Markup Language*)[22] é uma linguagem de marcação utilizada para descrever e estruturar dados para aplicações de *software*. As *tags* nessa linguagem não são pré-definidas, possibilitando ao desenvolvedor criar suas próprias marcações. Recomendada pelo W3C, essa linguagem possui os seguintes objetivos (W3C, 2014):

a) XML deve ser diretamente utilizável através da Internet;

b) Suportar uma ampla variedade de aplicações;

c) Ser compatível com SGML (*Standard Generalized Markup Language*) [23];

d) XML deve fornecer conteúdo que são legíveis por pessoas;

e) O projeto XML deve ser formal e conciso;

f) Documentos XML devem ser fáceis de criar.

De acordo com o que é discutido nas Seções 2.5 e 2.6, XML é utilizado em alguns padrões da Web Semântica como, por exemplo, OWL e RDF (*Resource Description Framework)*[24]. A Listagem 2-1 contém um exemplo de um documento XML.

[22] http://www.w3schools.com/xml/

[23] http://www.w3.org/MarkUp/SGML/

[24] http://www.w3.org/RDF/

Listagem 2-1 Exemplo de um documento XML

```
1.<?xml version="1.0" encoding="UTF-8”? >
2.<pesquisa instituicao="Unifacs">
3.<dissertacao autor="Cleber Lira" data="2014-09-11">
4.<orientador nome="Paulo" />
5.<tema>Servicos Web Semanticos</tema>
6.<conteudo>
7.      Exemplo simples do documento XML
8.</ conteudo >
9.</dissertacao>
10.</pesquisa>
```

Um documento XML é formado por elementos. Esses elementos são definidos por *tags* que estabelecem o início e o fim de seu escopo conforme é demonstrado na Listagem 2.1.

Documentos XML podem possuir seus elementos definidos por diferentes esquemas. Documentos WSDL, por exemplo, que definem a interface de um *Web Service* e serão discutidos na Seção 2.4 são definidos por um esquema, e os tipos de dados que são usados nas mensagens dos serviços são definidos por outros esquemas. Para evitar conflitos de nome em elementos e atributos são utilizados *namespaces*[25].

2.1.2 *XML Schema*

XML Schema é um padrão definido pelo W3C, baseado em XML, que possui como objetivo descrever a estrutura de um documento XML. O padrão *XML Schema* também é conhecido como *XML Schema Definition* [26]. A Listagem 2-2 ilustra a descrição da estrutura do documento XML definido na Listagem 2-1.

[25] http://www.w3schools.com/xml/xml_namespaces.asp

[26] http://www.w3schools.com/schema/default.asp

Listagem 2-2 Exemplo de *XML Schema*

```
1.<? xml version="1.0" encoding="UTF-8”?>
2.<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
3.xmlns:msg="http://www.unifacs.br/dissertacoes"
4.targetNamespace="http://www.unifacs.br/dissertacoes ">

5.<xs:element name="pesquisa">
6.   <xs:complexType>
7.      <xs:sequence>
8.         <xs:element ref="pesq:dissertacao" maxOccurs="unbounded"/>
9.         <xs:element name="orientador" type="xs:string"/>
10.          <xs:element name="tema" type=" xs:string "/>
11.       <xs:element name=" conteudo " type=" pesq: tpConteudo "/>
12.     </xs:sequence>
13.      <xs:attribute name="instituicao" type="xs:string" use="required"/>
14.  </xs:complexType>
15.</xs:element>

16.<xs:element name="dissertacao">
17.    <xs:complexType>
18.       <xs:attribute name="autor" type="xs:string" use="required"/>
19.       <xs:attribute name="data" type="xs:date" use="required"/>
20.    </xs:complexType>
21.</xs:element>

22.<xs:complexType name="tpConteudo">
23.   <xs:sequence>
24.      <xs:any />
25.   </xs:sequence>
26.</xs:complexType>
```

Um documento *XML Schema* é composto por declarações de tipos, elementos, atributos e a cada elemento é associado um tipo. Essa associação pode ser realizada de duas maneiras:

a) Declarando o tipo juntamente com o elemento, e.g. nos elementos pesquisa e dissertacao;

b) Utilizando o atributo *type* para referenciar um tipo declarado separadamente. A declaração do tipo pode ser realizada no mesmo documento como, por exemplo, *tpConteudo* ou importado de um outro documento XML *Schema Definition* como *xs:string*.

Conforme apresentado na Seção 2.7, o documento XML *Schema* poderá ser anotado semanticamente e utilizado durante a requisição de um determinado *Web Service*.

2.2 WSDL

A WSDL é uma linguagem para definição de interface baseada em XML utilizada para descrever uma funcionalidade oferecida por um *Web Service*. De acordo com Jewell e Chappell (2002) a WSDL se diferencia de outras linguagens de descrição de interface, como por exemplo, IDL (*Interface Definition Language*) [27] para descrever sistemas CORBA da OMG (OMG, 2014), pela sua característica de extensibilidade, que possibilita:

a) Agrupar conjuntos de operações como um tipo abstrato, que posteriormente pode ser mapeado para um protocolo e tipo de dados concreto;

b) Descrever pontos de acesso e suas mensagens, sem se preocupar com o formato da mensagem ou protocolo de rede utilizado;

c) Tratar mensagens como descrições abstratas dos dados sendo trocados.

A linguagem WSDL foi desenvolvida pela IBM [28], Microsoft [29] e Ariba [30] através da combinação de duas linguagens para descrição de serviço: NASSL (*Network Application Service Specification Language*) da IBM e SDL (*Service Description Language*) da Microsoft.

WSDL é uma especificação do W3C e atende a necessidade de descrever as comunicações que ocorrem na web de uma forma estruturada, pois utiliza uma gramática XML para descrever os serviços. De acordo com Prazeres (2009) a WSDL é utilizada para descrever um *Web Service* especificando sua localização e descrevendo as suas operações. Os documentos WSDL estabelecem um contrato no qual são descritas as restrições, os requisitos técnicos e informações necessárias para o consumidor utilizar o *Web Service*.

---

[27] http://documentation.progress.com/output/Iona/orbix/gen3/33/html/orbix33java_pguide/IDL.html

[28] http://www.ibm.com/br/pt/

[29] http://www.microsoft.com/pt-br/default.aspx

[30] http://www.ariba.com/

2.2.1 Estrutura de um documento WSDL 2.0

O documento WSDL 2.0 descreve um *Web Service* em duas seções: uma abstrata e outra concreta conforme ilustrado na Figura 2-1. Em cada seção, é utilizado um número de construções para promover a sua reutilização e a separação de responsabilidades no projeto. Em um nível abstrato, o serviço é descrito de acordo com as mensagens que envia e recebe. Essas mensagens são descritas independentes de um formato específico. Em um nível concreto, é especificado, por exemplo, o protocolo de transporte utilizado e os detalhes de uma ou mais interfaces do serviço. Um *Web Service* é definido em WSDL 2.0 pelos seguintes elementos (JENKOV, 2014):

a) *description*: É o elemento raiz de um documento WSDL 2.0;

b) *types*: contém a especificação dos tipos de dados trocados entre o cliente e o serviço web. Por padrão, esses tipos de dados são descritos utilizando *XML Schema*;

c) *interface*: descreve as operações que o serviço possui e quais mensagens são trocadas a cada operação (input/output) bem como as possíveis mensagens de falha;

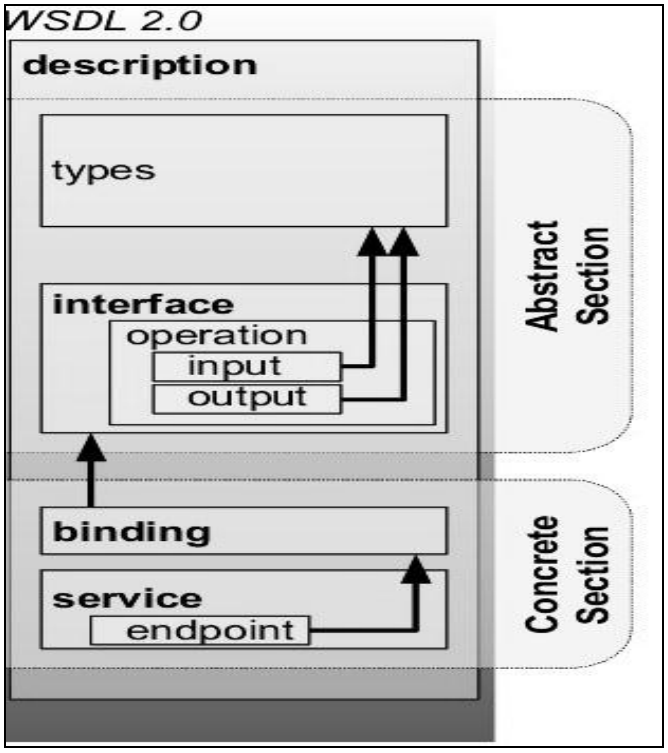
d) *binding*: esse elemento descreve como o *Web Service* é acessado através da rede. Normalmente o elemento *binding* conecta o serviço web com o protocolo HTTP;

e) *service*: determina o local onde o *Web Service* pode ser acessado na rede, normalmente, através de uma URL;

f) *documentation*: é um elemento opcional e pode ser utilizado para fornecer uma descrição legível do serviço para os desenvolvedores;

g) *import*: elemento opcional e pode ser utilizado para importar esquemas XML ou outros arquivos WSDL.

Figura 2-1 Estrutura de um documento WSDL 2.0



Fonte: Adaptado do W3C (2007).

De acordo com Mandel (2008), o documento WSDL 2.0 é utilizado para descrever sintaticamente Serviços Web baseado no padrão SOAP sendo também uma recomendação do W3C para a descrição de Serviços Web *restful*. A Listagem 2-3 mostra a descrição de um *Web Service restful* descrito sintaticamente na linguagem WSDL.

Listagem 2.3 Descrição de um *Web Service restful*

```
1. <wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
2.   targetNamespace="http://www.bookstore.org/booklist/wsdl"
3.   xmlns:tns="http://www.bookstore.org/booklist/wsdl"
4.   xmlns:whttp="http://www.w3.org/ns/wsdl/http"
5.   xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions"
6.   xmlns:xs="http://www.w3.org/2001/XMLSchema"
7.   xmlns:msg="http://www.bookstore.org/booklist/xsd">
8.  <wsdl:documentation>
9.       This is a WSDL 2.0 description of a sample bookstore service listing for obtaining book
10.information.
11.  </wsdl:documentation>
12.  <wsdl:types>
13.    <xs:import namespace="http://www.bookstore.org/booklist/xsd"
```

```
14.      schemaLocation="booklist.xsd"/>
15.  </wsdl:types>
16.  <wsdl:interface name="BookListInterface">
17.    <wsdl: operation name="getBookList"
18.      pattern="http://www.w3.org/ns/wsdl/in-out"
19.      style="http://www.w3.org/ns/wsdl/style/iri"
20.      wsdlx:safe="true">
21.     <wsdl:documentation>
22.       This operation returns a list of Books
23.     </wsdl:documentation>
24.     <wsdl: input element="msg:getBookList"/>
25.     <wsdl: output element="msg:bookList"/>
26.   </wsdl: operation>
27.  </wsdl:interface>
28.  <wsdl:binding name="BookListHTTPBinding"
29.    type="http://www.w3.org/ns/wsdl/http"
30.    interface="tns:BookListInterface">
31.   <wsdl:documentation>
32.     The RESTful HTTP binding for the book list service.
33.   </wsdl:documentation>
34.   <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
35.  </wsdl:binding>
36.  <wsdl:service name="BookList" interface="tns:BookListInterface">
37.   <wsdl:documentation>
38.     The bookstore's book list service.
39.   </wsdl:documentation>
40.   <wsdl:endpoint name="BookListHTTPEndpoint"
41.     binding="tns:BookListHTTPBinding"
42.     address="http://www.bookstore.com/books/">
43.   </wsdl:endpoint>
44.  </wsdl:service>
45. </wsdl:description>
```

Fonte: Mandel (2008).

2.3 SERVIÇOS

De acordo com Fugita e Hirama (2012) um serviço é um módulo de *software* que possui a capacidade de realizar uma função específica quando requisitado por seus

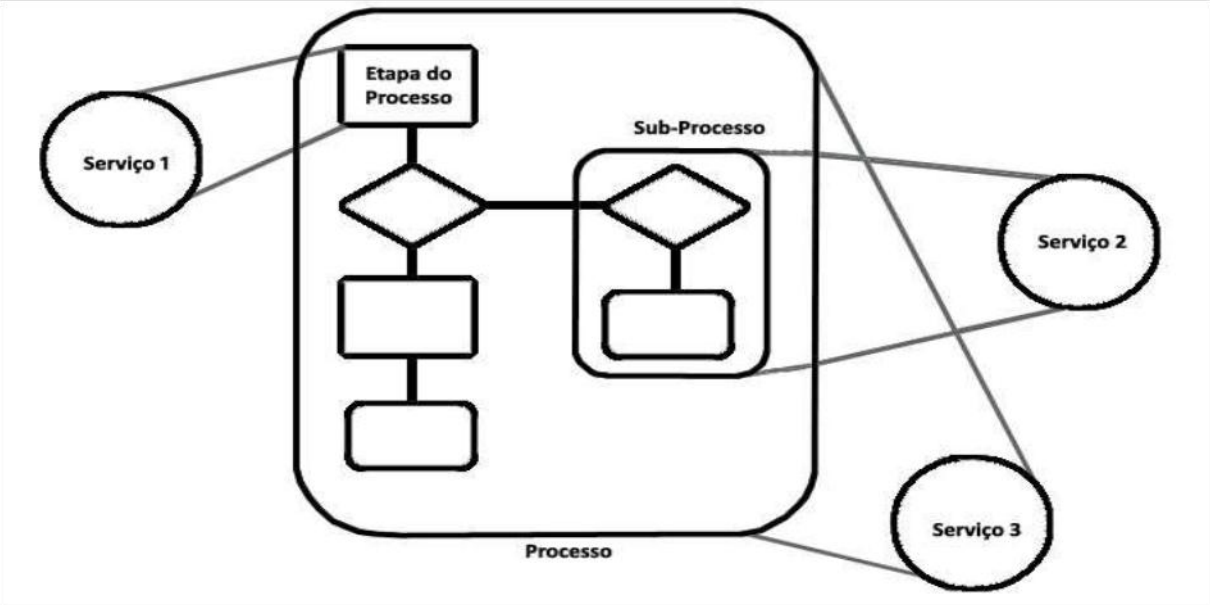
consumidores. Para Erl (2011), os serviços são construídos a partir de unidades lógicas individuais e autônomas que são estruturadas por processos de negócios com o objetivo de solucionar problemas. Borges (2012), afirma que essas unidades lógicas devem aderir aos princípios e padrões previamente estabelecidos, por exemplo, uma linguagem comum para comunicação em uma organização. Essa padronização facilita a solicitação dos serviços pelos consumidores.

Fugita e Hirama (2012) afirmam que um serviço encapsula diversas operações para atender a uma tarefa específica de um processo de negócio. Por sua vez, um processo de negócio é um agregado de diversos serviços que se relacionam entre si. A associação entre esses serviços é baseada no entendimento da interação que um serviço tem com o outro, a partir da descrição do serviço e da troca de informações baseadas em mensagens.

A Figura 2-2 ilustra serviços que encapsulam operações lógicas em diferentes etapas de um processo de negócio:

a) O “Serviço 1” está encapsulando uma etapa do processo;
b) O “Serviço 2” possui um sub-processo que contém várias etapas;
c) O “Serviço 3” é composto pelos “Serviço 1” e “Serviço 2” representando todo o processo de negócio.

Figura 2-2 Serviços encapsulando operações lógicas de diferentes tamanhos



Fonte: Erl (2011).

De acordo com Erl (2009) os componentes (serviços, descrição e mensagens) que compõe o núcleo da arquitetura de um serviço possuem as seguintes características:

a) baixo acoplamento: fraca dependência na comunicação entre os serviços;
b) contrato: representado por um documento WSDL conceituado na Seção 2.2;
c) abstração: com a capacidade de mitigar a complexidade do processo;
d) capacidade de reuso: possibilita condições para que diferentes ambientes possam compartilhar os mesmos recursos;
e) autonomia: faz referência ao nível de controle que um serviço tem sobre seu ambiente e recurso;
f) visibilidade: facilidade de identificação para fornecer a oportunidade de reuso;
g) manutenção do estado: estado deve ser mantido apenas quando for necessário;
h) composição: atuar como membro de outros serviços.

Os serviços são disponibilizados em repositórios nos quais são incluídos as documentações, o código fonte e os demais artefatos que envolvem o serviço (BORGES, 2012). Os serviços podem ser implementados por intermédio de qualquer linguagem de programação possibilitando o reuso a partir de diferentes aplicações (SOMMERVILLE, 2007).

Erl (2011) afirma que a automatização dos serviços pode ser realizada por meio da tecnologia de Serviços Web que será conceituada na Seção 2.4.

2.4 SERVIÇOS WEB

De acordo com o W3C, um *Web Service* é um sistema de *software* que oferece um serviço projetado para suportar a interação entre as máquinas na rede, tais como a Internet/Intranet. Conforme Kalin (2010), os Serviços Web podem ser classificados em dois grupos: SOAP e REST.

2.4.1 SOAP

SOAP é um protocolo de comunicação baseado em XML que possibilita a troca de informações entre Serviços Web. De acordo com o W3C (2007) essa comunicação poderá ocorrer tanto em ambientes centralizados como em ambientes distribuídos. Esse protocolo possui suas definições estabelecidas em um documento WSDL localizada na seção concreta, especificamente no elemento *binding*, que foi discutido na Seção 2.2.

O protocolo SOAP para Kumar, Narayan e Ng (2010), é similar e uma evolução de outros protocolos de rede como o IIOP (*Internet Inter - ORB Protocol*) [31] para o CORBA, o ORPC (*Object Remote Procedure CALL)*[32] para o DCOM e o JRMP (*Java Remote Method Protocol*) [33] para o RMI (*Remote Method Invocation*) [34], pois possibilitam que aplicativos possam trocar informações em um ambiente distribuído e de uma maneira descentralizada. Os autores afirmam que o SOAP foi projetado independentemente do protocolo de transporte, sendo assim, uma mensagem SOAP pode ser transportada em qualquer protocolo sem alterar a estrutura e o conteúdo de sua mensagem. A Listagem 2-4 exemplifica a estrutura de um documento SOAP. A Listagem 2-4 apresenta a estrutura de um documento SOAP constituído pelos seguintes elementos:

a) Envelope

Esse é o elemento raiz, que identifica o documento XML como sendo uma mensagem SOAP.

b) *Header*

Esse elemento é opcional e contém informações específicas do aplicativo da mensagem SOAP. Caso este elemento esteja presente, ele deverá ser o primeiro elemento filho do elemento Envelope.

c) *Body*

Esse elemento armazena informações de requisição e resposta, ou seja, o *body* contém as mensagens pretendidas pela aplicação cliente.

d) *Fault*

Esse elemento armazena registros de possíveis falhas que poderão ocorrer durante a comunicação entre as aplicações.

De acordo com Damasceno (2009), os Serviços Web podem ser agrupados em quatro camadas, conforme ilustrado na Figura 2-2:

a) camada de pesquisa, responsável pelo registro da Serviços Web em repositórios, contendo os protocolos (UDDI[35] e EbXML[36]);

---

[31] http://publib.boulder.ibm.com/infocenter/realtime/v1r0/index.jsp?topic=%2Fcom.ibm.rt.doc.10%2Frmi-iiop%2Foverview.html

[32] http://msdn.microsoft.com/en-us/library/cc226906.aspx

[33] http://www.javaworld.com/article/2076547/soa/rmi-over-iiop.html

[34] http://www.oracle.com/technetwork/articles/javaee/index-jsp-136424.html

[35] http://searchsoa.techtarget.com/definition/UDDI

b) camada de descrição do serviço, contendo o documento WSDL;

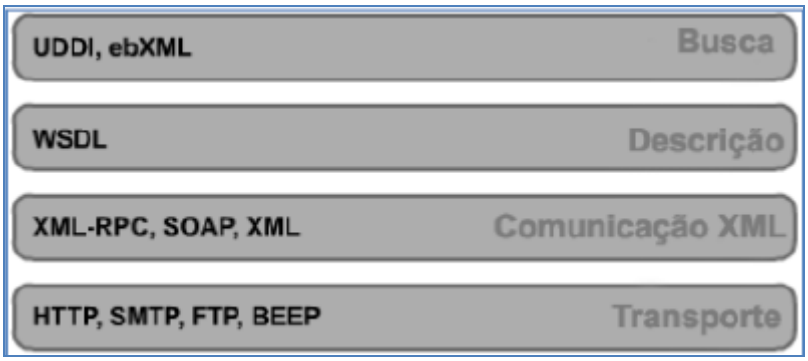
c) camada de comunicação baseada em XML, contendo o protocolo SOAP;

d) camada de transporte, que pode conter protocolos como HTTP, SMTP, FTP e BEEP[37].

Listagem 2-4 Estrutura de um documento SOAP

```
1.<?xml version="1.0”? >
2.<soap: Envelope
3.xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
4.soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
5.<soap: Header>
6. ...
7.</soap:Header>
8.<soap: Body>
9. ...
10. <soap:Fault>
11. ...
12. </soap:Fault>
13. </soap:Body>
14.</soap:Envelope>
```

Figura 2-3 Camadas de protocolo dos Serviços Web SOAP



Fonte: Damasceno (2009).

[36] http://www.ebxml.org/

[37] http://www.beepcore.org/

2.4.2 REST

REST é conhecido como um estilo arquitetural para facilitar o reuso das aplicações em ambientes distribuídos, mantendo entre suas características interfaces generalizadas, interação entre componentes com fácil escalabilidade e uma instalação que não depende destes componentes. Nesse estilo arquitetural, as mensagens que são trocadas entre os aplicativos são encapsuladas diretamente no protocolo HTTP. As requisições no ambiente REST são feitas para recursos em vez de serem realizadas para serviços e diferentemente de outras abordagens as chamadas não são realizadas para procedimentos. Cada recurso é representado por uma URI. O REST atualmente é usado com o protocolo HTTP devido a escalabilidade deste protocolo, garantindo uma grande visibilidade e quantidade de componentes e aplicativos com capacidade de se comunicar com os serviços criados com base nessa arquitetura. Existem quatro princípios que regem a arquitetura REST, os quais são descritos a seguir (FIELDING, 2000):

a) Sintaxe universal

Este princípio também é denominado de endereçabilidade, pois permite fornecer uma identificação dos recursos através de URIs. Assim, cada recurso possuirá um endereço que poderá ser utilizado na descoberta dos serviços. Pode-se afirmar que o endereço http://www.unifacs.br/main/capa/default.aspx é um URI válido para o recurso "Universidade Salvador".

b) Interface Uniforme

Este princípio refere-se ao conjunto de operações: inclusão, alteração, remoção e consulta que manipulam os recursos. Cada operação utiliza um método diferente do protocolo HTTP:

- O método PUT é responsável em incluir um novo recurso;
- O método POST é responsável em atualizar o recurso;
- O método DELETE é responsável em excluir o recurso;
- O método GET retorna o estado atual do recurso.

Diante dessa característica, se o consumidor possuir acesso aos recursos oferecidos por um determinado serviço, automaticamente saberá como incluir, alterar, remover e consultar tais recursos.

c) Mensagens auto descritivas

Os recursos podem ser acessados em diferentes formatos (e.g. JSON[38], XML, PDF[39]). Desta forma, os recursos não possuem uma associação com suas representações.

d) Estado Não-Persistente

A comunicação realizada com os recursos ocorre sem o armazenamento do estado quando utilizado com o protocolo HTTP. Caso haja necessidade de manutenção do estado, existem técnicas como a utilização de *cookies*[40] que podem ser empregadas para tal finalidade.

Conforme Ferreira (2011), Serviços Web que seguem o padrão REST costumam ser implementados através de duas abordagens:

a) Hi – REST

Essa é a abordagem descrita por (FIELDING, 2000). Os Serviços Web implementados com base em uma arquitetura REST são denominados de Serviços Web *restful* (RICHARDSON; RUBY, 2008). De acordo com Ferreira (2009), foram Richardson e Ruby (2008) os responsáveis em conectar REST com o protocolo HTTP, fornecendo um aspecto tecnológico aos quatro princípios discutidos anteriormente e estabelece ROA (*Resource-Oriented Architecture*)[41] que é uma arquitetura que segue os princípios REST.

b) Lo-REST

De acordo com Xavier (2011), essa abordagem utiliza apenas os métodos GET e POST do protocolo HTTP para realizar as operações. Ainda segundo o autor, isso ocorre porque alguns *proxies* e *firewalls* não permitem realizar os outros métodos (e.g.: PUT, DELETE) do protocolo HTTP. De acordo com Pautasso, Zimmermann e Leymann (2008) para identificar uma operação (inclusão, alteração ou exclusão) pode ser passada, na solicitação, uma variável para identificação.

2.4.3 Comparação entre REST e SOAP

Para Tyagi (2006), devido ao maior número de camadas e protocolos, Serviços Web SOAP consomem maior banda de rede que Serviços Web *restful*. De acordo com Xavier

[38] http://jsonformat.com/

[39] http://www.adobe.com/br/products/acrobat/adobepdf.html

[40] http://www.w3schools.com/js/js_cookies.asp

[41] https://books.google.com.br/books?id=XUaErakHsoAC&printsec=frontcover&hl=pt-BR#v=onepage&q&f=false

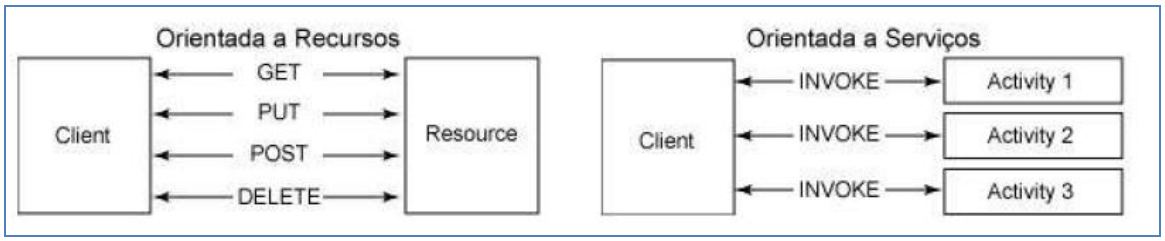
(2011), Serviços Web construídos utilizando o protocolo SOAP possuem várias camadas recomendando a utilização de REST em aplicações que serão acessadas a partir de dispositivos *mobile* ou de maneira *ubíqua*[42].

De acordo com Maleshkova, Pedrinaci e Domingue (2009) Serviços Web *restful* possuem limitações em comparação aos Serviços Web SOAP por não existir uma linguagem estruturada para descrever esses serviços amplamente aceita por desenvolvedores de *software*. Entretanto, Mandel (2008) afirma que é possível descrever Serviços Web *restful* em documentos WSDL 2.0 e segundo Hadley (2006) é possível descrever também esses Serviços Web em documentos WADL (*Web Application Description Language*)[43].

Maleshkova, Pedrinaci e Domingue (2009) afirmam que muitas aplicações da Web 2.0 expõe suas funcionalidades através de Serviços Web *restful*, isso ocorre porque muitas dessas aplicações serão acessadas através de dispositivos móveis e conforme afirmado anteriormente Serviços Web SOAP consomem muita banda de rede.

A Figura 2-4 ilustra as interfaces dos Serviços Web que são implementados nas abordagens REST e SOAP. Em REST, o acesso aos recursos ocorre através dos métodos do protocolo HTTP enquanto nos Serviços Web SOAP o acesso ocorre através das chamadas a procedimentos. Conforme Pautasso, Zimmemann e Leymann (2008) a vantagem dos Serviços Web *restful* é a utilização das URIs para identificar e localizar os recursos pois determina um endereço global não havendo a necessidade de centralizar a chamada a um procedimento.

Figura 2-4 Abordagem orientada a recursos x Abordagem orientada a serviços



Fonte: Snell (2004).

2.5 RDF/RDF SCHEMA

RDF (*Resource Description Framework*)[44] é um *framework* para representar a informação sobre os recursos na Internet (KLYNE; CARROLL, 2006). Os recursos podem ser qualquer objeto que incluem: (i) documentos, (ii) conceitos abstratos, (iii) números, (iv)

[42] http://www.scielo.br/scielo.php?pid=S1415-65552007000400009&script=sci_arttext

[43] https://wadl.java.net/wadl20051116.pdf

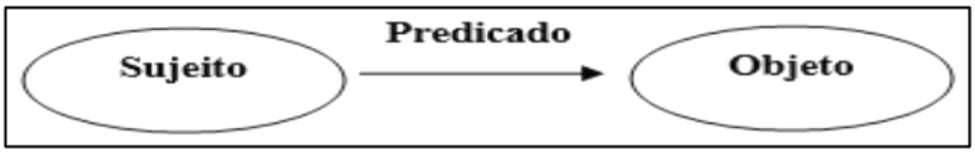
[44] http://www.w3.org/RDF/

strings, (v) arquivos e que possam ser identificados na web, como por exemplo, acesso às informações em um banco de dados ou informações acerca do perfil do usuário em uma rede social *online* (W3C, 2014).

RDF possui como objetivo descrever os recursos com informação que possa ser interpretada por aplicações de *software*, e não apenas interpretadas por seres humanos (XAVIER, 2011). De acordo com o W3C (2014), cada recurso é identificado por um IRI (*Internationalized Resource Identifier*)[45]. IRI é uma generalização do URI e possibilita a utilização de caracteres *Unicode*[46] (W3C, 2014).

Os recursos são descritos em termos de propriedades dos elementos, relacionamentos entre esses elementos e os próprios recursos que são realizados através de triplas sendo representados pela utilização de grafos. Uma tripla é composta pelo conceito de sujeito, predicado e objeto. Uma página Web, por exemplo, é denominada de sujeito (recurso); informações relacionadas ao desenvolvedor da página é denominada de predicado (propriedade) e a parte que determina o valor da propriedade é denominada objeto. O IRI, http://www.unifacs.br tem um desenvolvedor cujo valor é "Fábrica de Software" que poderia ser representada http://www.unifacs.br por uma declaração RDF como: (i) o sujeito é o IRI, http://www.unifacs.br; (ii) o predicado é a palavra desenvolvedor; (iii) e o objeto é a palavra Fábrica de *Software*. A Figura 2-5 ilustra um grafo que ilustra o conceito de tripla. Nesta representação, sujeito e objeto são recursos e o predicado é o relacionamento semântico entre eles. O sujeito, o predicado e o objeto são representados por IRIs.

Figura 2-5 Grafo RDF com dois vértices (sujeito e objeto) e um arco (predicado)



Fonte: Baseado W3C (2014).

RDFS (*Resource Description Framework Schema*)[47] é uma extensão do RDF que fornece um vocabulário para descrever recursos e propriedades (BRICKLEY; GUHA; MCBRIDE, 2014). A classe e propriedade do sistema RDF *Schema* é semelhante aos

[45] http://www.w3.org/2011/rdf-wg/wiki/IRIs/RDFConceptsProposal

[46] http://www.unicode.org/

[47] http://www.w3.org/TR/rdf-schema/

sistemas de tipos de linguagens de programação orientada a objeto, como por exemplo C#, possibilitando criar o conceito de herança de recursos com a utilização do rdfs: subClassOf e propriedade com o uso de rdfs:subPropertyOf. Para Xavier (2011) ao fazer uso de RDF e RDFS é possível construir modelos de representação do conhecimento bem elaborados. O autor afirma que ao utilizar a propriedade rdf:type é possível definir um indivíduo (instância) a partir de uma classe. Por esse motivo, em uma tripla contendo essa propriedade, o objeto e o sujeito podem ser uma classe, uma propriedade ou indivíduo. Dessa forma é possível criar indivíduos (instâncias) de outros indivíduos e propriedades (instâncias) de outras propriedades. Existem muitos tipos de conhecimentos que não podem ser expressos por RDFS. Leenheer (2009) descreve algumas limitações do RDFS para descrever os recursos:

a) Não existem restrições de existência e cardinalidade: não é possível estabelecer, por exemplo, que todas as instâncias de Veículo possuem um pai que também é Veículo, ou que todas os Veículos possuem exatamente um dono.

b) Não existem propriedades transitivas, simétricas ou inversas: não é possível estabelecer, por exemplo, que A é uma propriedade transitiva, que B é o seu inverso ou que elas são simétricas.

c) Não existem restrições localizadas de domínio e âmbito: não é possível, por exemplo, afirmar que a propriedade possui Roda terá como domínio Carro quando o âmbito for Carro e Moto quando o âmbito for Moto.

2.6 OWL

A *Web Ontology Language* (OWL) (OWL 2, 2012) [48] é uma linguagem definida pela W3C que descreve como projetar ontologias (GRUBER, 1993). Trata-se de uma especificação que permitem as aplicações processarem o conteúdo da informação em vez de realizar apenas a visualização da informação.

Conforme Cardoso e Pinto (2015) a OWL fornece uma maior interoperabilidade para as aplicações, pois possibilita uma maior interpretação pelos aplicativos que processa conteúdo web do que RDF, RDFS e XML. Ela oferece um vocabulário adicional que permite descrever as relações entre as classes e as características das propriedades.

[48] http://www.w3.org/TR/owl2-overview/

Sendo baseado em XML, esta linguagem permite facilmente a troca de informações entre diferentes tipos de dispositivos usando os mais diversos tipos de linguagens de programação e sistemas operacionais, ou seja, em um ambiente extremamente heterogêneo.

A OWL fornece três tipos de sublinguagens que foram projetados especificamente para diferentes tipos de comunidades de desenvolvedores e usuários (W3C, 2004):

a) OWL *LITE*:

É uma linguagem derivada da OWL que utiliza algumas de suas características e possui mais limitações que a OWL DL e OWL FULL.

b) OWL DL (*Description Logic*):

Fornece suporte aos usuários que necessitam obter uma maior expressividade, entretanto, mantendo todo o aspecto computacional e permitindo que todo o processo de informação seja executado em tempo finito. A sigla DL fornece ênfase à lógica descritiva, que é uma área de pesquisa que estuda a lógica de primeira ordem (BAADER, 2003).

c) OWL *FULL*:

Fornece suporte aos usuários que necessitam de uma maior expressividade sem garantias computacionais e alinhados a liberdade sintática do RDF.

De acordo com o W3C (2004) cada uma destas linguagens é uma extensão da sua predecessora, por exemplo, uma ontologia válida em OWL *Lite* é uma ontologia válida em OWL DL que por sua vez é uma ontologia válida em OWL *Full*.

De acordo com Cardoso e Pinto (2014) a partir de 2009, o W3C recomendou OWL 2 como linguagem padrão para definição de ontologias. Essa nova especificação estabeleceu o conceito de perfil que é similar ao conceito das linguagens derivadas da OWL. Segundo Torres (2012) existem três perfis:

a) OWL 2 EL

Utilizado em aplicações que faça uso de ontologias que empregam um grande número de propriedade e/ou classes. A sigla *EL* refere-se à base do perfil da família de lógicas de descrição.

b) OWL 2 QL

Utilizada em aplicações que consultam grandes volumes de dados, sendo de fundamental importância devolver resultados para as consultas. A sigla *QL*, está relacionado

ao fato que *request-response*, neste perfil, pode ser implementada utilizando uma linguagem de consulta relacional.

c) OWL 2 RL

É destinado a aplicações que exigem escalabilidade sem sacrificar o poder expressivo da linguagem. Ela é projetada para aplicações OWL que podem trocar a expressividade completa da linguagem pela eficiência e aplicações RDF que precisam de alguma expressividade a partir do uso da OWL 2.

A linguagem OWL define as ontologias que determinarão o vocabulário semântico dos Serviços Web. Conforme será discutido na Seção 2.7, o mecanismo SAWSDL utiliza ontologias para definir a semântica em documentos WSDL e *Xml Schema.*

2.7 SAWSDL

De acordo com (KOPECKY et al., 2007), para realizar a anotação semântica, a SAWSDL define atributos de extensão que podem ser aplicados tanto em elementos WSDL quanto em elementos do XML *Schema Defintion*. Os autores ainda afirmam que SAWSDL não possui nenhuma semântica especifica, pois possibilitam apenas anotar semanticamente documentos WSDL através de conceitos definidos em uma ontologia. Prazeres (2009) afirma que, SAWSDL diferencia-se de outras abordagens, pois realiza a anotação semântica dos Serviços Web no próprio documento WSDL. Por essa razão, o documento WSDL terá a capacidade de descrever as funcionalidades sintáticas bem como semânticas dos Serviços Web

Para Klusch, Kapanhanke e Zinnikus (2012) SAWSDL foi projetada como uma extensão da WSDL, pois permite aos desenvolvedores de Serviços Web enriquecer as descrições dessas aplicações incluindo informação semântica. Especificamente SAWSDL define dois atributos de extensão que são utilizados na definição da anotação semântica e que são descritos nos itens I e II e ilustrados na Figura 2-6 (KOPECKY et al., 2007):

I. *modelReference*

utilizado para especificar a associação entre elementos de um documento WSDL e/ou elementos de um XML *Schema Defintion* em algum modelo semântico. Essa associação é realizada através de uma ou mais IRIs, ou seja, é possível associar um elemento a mais de um modelo semântico. O *modelReference* pode ser utilizado, por exemplo, para auxiliar a determinar se determinado *Web Service* atende aos requisitos de um cliente. Esse atributo de extensão pode anotar:

- Elementos WSDL
  1) *Interfaces*
  2) *Operation*
  3) *Faults*
- Definições de tipo WSDL
  1) Definições de tipo complexo no *XML Schema*
  2) Definições de tipo simples
  3) Declarações de elementos
  4) Declarações de atributos

A Listagem 2-5 exemplifica o uso do atributo *modelReference* para anotar um documento WSDL, especificamente o elemento *operation*. As linhas 1,4,5,6 representam elementos do documento WSDL enquanto a linha 2 possui um IRI que aponta para a ontologia responsável em determinar o vocabulário semântico do elemento *operation* do documento WSDL.

Listagem 2-5 Utilizando o *modelReference* para anotar uma operação

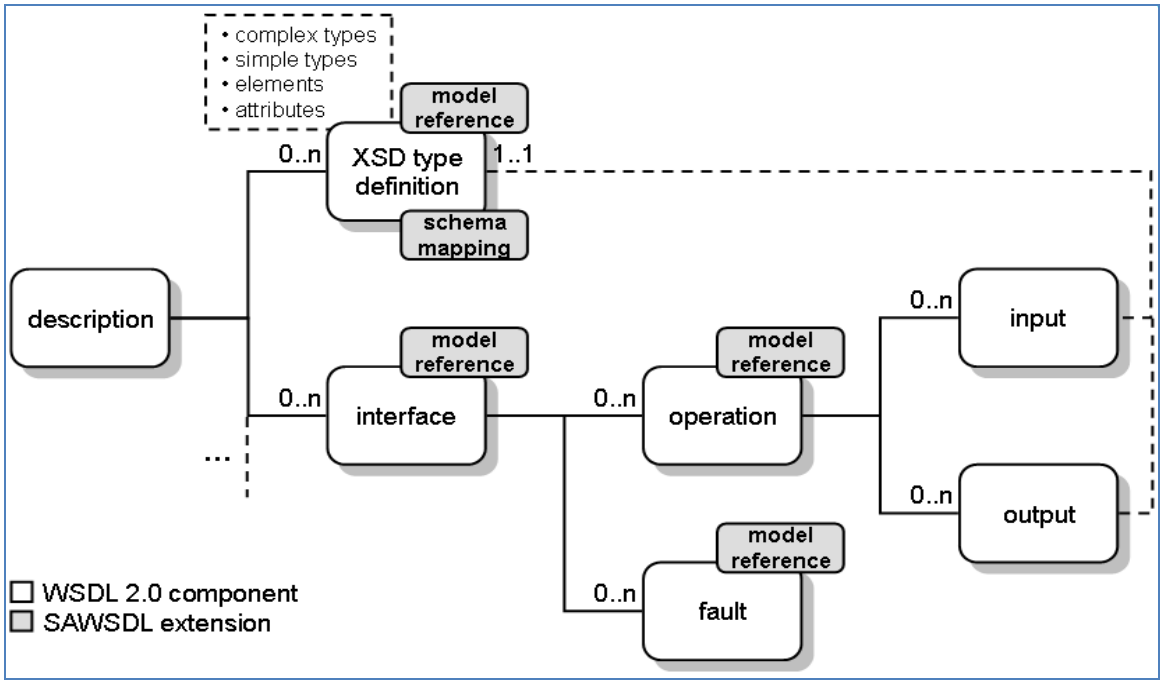
```
1.<wsdl:operation name="order"
2.sawsdl:modelReference=“http://www.w3.org/2002/ws/sawsdl/spec/ontology/rosetta#Reque
3.stPurchaseOrder">
4.<wsdl:input element="OrderRequest"/>
5.<wsdl:output element="OrderResponse"/>
6.</wsdl:operation>
```

A Listagem 2-6 exemplifica o uso do atributo *modelReference* para anotar um elemento de tipo simples (*simpleType*).

Listagem 2-6 Utilizando o modelReference para anotar um tipo simples.

```
1.<xs: simpleType name="confirmation"
2.sawsdl:modelReference=“http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseo
3.rder#OrderConfirmation">
4.   <xs:restriction base="xs:string">
5.       <xs:enumeration value="Confirmed"/>
6.      <xs:enumeration value="Pending"/>
7.       <xs:enumeration value="Rejected"/>
8.  </xs: restriction>
9.</xs: simpleType>
```

Figura 2-6 Atributos SAWSDL aplicados a elementos de documentos WSDL



Fonte: Klusch e Kapahnke (2008).

II. SchemaMapping

Dois atributos de extensão, denominados *liftingSch*emaMapping e *loweringSchemaMapping*, que são adicionados a declarações de elemento do esquema XML e definições de tipo para especificar mapeamentos entre os dados semânticos e o XML. Oliveira (2009) afirma que esses atributos são utilizados para realizar ajustes semânticos nas mensagens trocadas entre os Serviços Web. Esses ajustes possuem o objetivo de transformar a estrutura de dados especificada no documento WSDL para o formato exigido pelo modelo semântico associado e vice-versa.

Um mapeamento referenciado pelo atributo *liftingSch*emaMapping define como um documento XML em conformidade com o elemento ou tipo definido no esquema é transformado em dados que está de acordo com algum modelo semântico, isto é, o processo de transformação gerará como saída dados semânticos. A entrada para a transformação é o documento XML no qual a declaração de mapeamento está localizada. A Listagem 2-7 exemplifica esse mapeamento sendo realizado em um tipo simples (*simpleType*). Já o mapeamento referenciado pelo atributo *loweringSchemaMapping* define como os dados em um modelo semântico é transformado em XML. Nesse caso, a informação de entrada será o modelo semântico e o resultado do processo será o documento XML no

qual a declaração do mapeamento está localizada. Esses dois atributos de extensão podem anotar:

- *XML Schema Defintion*

1) Definições de tipo simples
2) Definições de tipo complexo

Listagem 2-7 Utilizando o *liftingSchemaMapping* para anotar um tipo simples

```
1.<xs:element name="OrderResponse" type="confirmation" />
2.<xs:simpleType name="confirmation"
3.sawsdl:modelReference="http://www.w3.org/2002/ws/sawsdl/spec/ontology/purchaseorder
4.#OrderConfirmation"
5.sawsdl:liftingSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/Respons
6.e2Ont.xslt">
7.  <xs:restriction base="xs:string">
8.    <xs:enumeration value="Confirmed" />
9.    <xs:enumeration value="Pending" />
10.    <xs:enumeration value="Rejected" />
11.  </xs:restriction>
12.</xs:simpleType>
```

Na Listagem 2-7, o mapeamento definido em "http://www.w3.org/2002/ws/sawsdl/spec/mapping/Response2Ont.xslt" será aplicado ao elemento *OrderResponse.* Nesse exemplo a linguagem de mapeamento é a XSLT (KAY, 2007). O Quadro 2-1 apresenta um resumo dos atributos de extensão fornecidos pela SAWSDL.

Quadro 2-1 Resumo da sintaxe fornecida pela SAWSDL

Nome	**Descrição**
ModelReference	Associa documentos WSDL e/ou XML *Schema Definition* em algum modelo semântico.
LiftingSchemaMapping	Transformação de um documento XML em dados semânticos.
LoweringSchemaMapping	Transformação de dados semânticos em documento XML.

2.8 SPARQL

SPARQL (*SPARQL Protocol and RDF Query Language*)[50] é uma linguagem para consulta semântica capaz de obter e manipular dados no formato RDF (W3C, 2008). De acordo com Xavier (2011) é possível realizar uma consulta *SPARQL* para diferentes repositórios semânticos através do *SPARQL Protocol*. Esse protocolo utiliza o WSDL 2.0 para descrever um meio de transmitir as consultas SPARQL para um serviço de processamento e devolver a resposta para a entidade que requisitou a consulta (CLARK; FEIGENBAUM; TORRES, 2008). A Listagem 2-8 exibe o exemplo de uma consulta (*SPARQL QUERY LANGUAGE*) e a Listagem 2-9 exibe a inclusão (*SPARQL UPDATE*) em um repositório utilizando *SPARQL*.

Listagem 2-8: Exemplo de uma consulta semântica

```
1.PREFIX foaf:    <http://xmlns.com/foaf/0.1/>
2.SELECT? nameX ?nameY ?nickY
3.WHERE
4. {?x foaf:knows ?y ;
5.      foaf:name ?nameX .
6.    ? y foaf:name ?nameY .
7.    OPTIONAL { ?y foaf:nick ?nickY }}
```

Fonte: Harris e Seaborne (2013).

Listagem 2-9: Exemplo de uma inclusão semântica

```
1.PREFIX dc: <http://purl.org/dc/elements/1.1/>
2. INSERT { <http://example/egbook> dc:title  "This is an example title" }
```

Fonte: Clark, Passant e Polleres (2013).

2.8.1 APACHE *Jena*

Jena é um *framework open source* construído em Java para implementação de aplicações na Web Semântica e *Linked Data*[51] com base nas recomendações do W3C para RDF e OWL. O *Jena* oferece uma API para SPARQL, RDF e OWL, além de possuir um componente para a construção de repositórios semânticos: o TDB[52] (APACHE, 2015).

[50] http://www.w3.org/TR/rdf-sparql-query/#introduction

[51] http://www.w3.org/standards/semanticweb/data

[52] https://jena.apache.org/documentation/tdb/

Xavier (2011), afirma que um repositório semântico é um sistema que realiza o armazenamento de triplas RDF possibilitando dessa forma a realização de consultas SPARQL para recuperação das informações. Utilizando *Jena*, o armazenamento e consultas das triplas RDF em arquivos são realizados utilizando o componente TDB.

No Capítulo 5 é exemplificado a definição do repositório semântico a partir da utilização do *framework Jena*.

2.9 CONSIDERAÇÕES FINAIS

O objetivo dos Serviços Web é oferecer um serviço que suporte a comunicação entre máquinas disponibilizadas em uma rede de computadores e que uma das principais tendências da web atual é a Web Semântica (PRAZERES, 2009), muitas aplicações disponibilizam suas interfaces como Serviços Web. Fornecer semântica a esses serviços possibilita que um computador ou agente de *software* possam determinar automaticamente o seu significado e implementar tarefas como (i) descoberta, (ii) requisição e (iii) composição automática de serviços. Budinoski et al (2010) conceituam essas tarefas:

a) descoberta automática de Serviços Web

Possibilita localizar automaticamente os Serviços Web para uma necessidade específica do usuário;

b) requisição automática de Serviços Web

Possibilita requisitar o serviço automaticamente utilizando um programa de computador ou um agente de *software* fornecendo apenas uma descrição declarativa desse serviço;

c) composição automática de Serviços Web

Possibilita a composição automática e a comunicação dos Serviços Web necessários para realizar uma tarefa complexa, no qual inclui a combinação de mais de um serviço para atender a determinada solicitação.

Este trabalho considera que Serviços Web *restful* descritos em documentos WSDL 2.0 devem ser anotados semanticamente através de uma ferramenta baseada em serviços. Sommerville (2007), afirma que em um ambiente de negócio de rápidas mudanças poderão ocorrer problemas no desenvolvimento de *software*, como por exemplo, o aumento do custo para construí-los e, por isso, Erl (2011) destaca a importância do desenvolvimento de sistemas empregando o conceito de Serviços, discutido na Seção 2.3, que favorece a construção de ferramentas adaptáveis a mudanças. Guttula (2012) afirma que são necessárias a criação de

ferramentas que possibilitem fornecer automação na realização da anotação semântica, pois realizar essa atividade de forma manual requereria várias horas de um desenvolvedor de *software*. Por esse motivo, este capítulo teve o objetivo de apresentar os conceitos que subsidiam a elaboração e entendimento desta dissertação.

A inclusão de semântica nos documentos *XML Schema Defintion*, discutidos na Seção 2.1, e em *Web Service restful*, discutido na Seção 2.4, descrito sintaticamente em documento WSDL 2.0, discutido na Seção 2.2, através da especificação SAWSDL que foi discutido na Seção 2.7, conduzido pela utilização de ontologias construídas em linguagem OWL, discutida na Seção 2.6, possibilitam que na Web Semântica as tarefas de requisição, descoberta e composição dessas aplicações passem a acontecer de forma automática. Serviços Web *restful* semânticos podem ser consultados utilizando a linguagem, baseada em RDF (discutido na Seção 2.5), SPARQL discutida na Seção 2.8.

No Capítulo 3, são discutidos alguns trabalhos pesquisados, relacionados com o contexto desta dissertação.

3 REVISÃO BIBLIOGRÁFICA PARA ANOTAÇÃO SEMÂNTICA EM SERVIÇOS WEB

Este capítulo discute os principais trabalhos encontrados na literatura sobre anotação semântica de Serviços Web e está estruturado da seguinte maneira: inicialmente, a Seção 3.1 apresenta como foi realizada a pesquisa bibliográfica para obtenção dos trabalhos que propõem meios de fornecer semântica em Serviços Web e são relacionados a esta dissertação. A Seção 3.2 discorre sobre os trabalhos considerados mais relevantes envolvendo considerações sobre tecnologias e padrões aceitos pela indústria de *software* na anotação de Serviços Web semânticos, especialmente as ferramentas que utilizam a abordagem REST. Serão consideradas propostas de ferramentas criadas para funcionar em ambiente web bem como soluções capazes de funcionar em um determinado ambiente de desenvolvimento integrado IDE (*Integrated Development Environment*)[53] e a Seção 3.3 que fornece uma discussão e considerações finais deste capítulo.

3.1 METODOLOGIA PARA A PESQUISA BIBLIOGRÁFICA

Uma pesquisa bibliográfica foi conduzida no intuito de encontrar e analisar trabalhos relacionados a anotação semântica de Serviços Web que pudessem responder as questões de pesquisa apresentadas no Capítulo 1. Os tópicos do protocolo que a guiou são abordados nesta seção.

3.1.1 Estratégia de busca

A formação das palavras chave (*strings de busca)* para a busca na web foi construída obedecendo a seguinte sequência:

a) Os termos foram identificados a partir das questões de pesquisa;

b) Foi realizada uma tradução dos termos identificados para o inglês;

c) Foi realizada pesquisa de busca dos termos relacionados aos tópicos identificados na investigação.

[53] http://en.wikipedia.org/wiki/Integrated_development_environment

Quadro 3-1 *Strings* de busca para o idioma português

Strings para busca
("Anotação Semântica de Serviços Web Restful") or ("Aplicação para anotação semântica de serviços web" or "Ferramentas para anotação semântica de serviços web ") or ("serviços para anotação semântica de serviços web")

Quadro 3-2 *Strings* de busca para o idioma inglês

Strings para busca no idioma inglês
("Semantic Annotation for Web Services Restfull ") or ("Application for semantic annotation of web services" or "Tools for semantic annotation of web services") or ("services for semantic annotation of web services")

3.1.2 Fontes de Pesquisa

Para seleção das fontes de pesquisa foram considerados:

a) a disponibilidade de consultar os artigos na web;

b) a presença de mecanismos de busca usando palavras;

c) a importância e relevância das fontes, considerando prioritários os publicados em congressos, revistas, artigos, dissertações e teses relacionadas aos tópicos identificados na investigação.

Foram utilizadas as seguintes fontes de pesquisa:

a) Google (http://www.google.com.br)

b) Google acadêmico (http://scholar.google.com.br)

c) ACM Digital Library (http://dl.acm.org/)

d) *IEEEXplore Digital Library (*http://ieeexplore.ieee.org/Xplore/home.jsp)

3.1.3 Resultados

A partir das *strings* nos idiomas português e inglês, as buscas primárias forneceram uma boa base para auxiliar no processo de identificação dos trabalhos similares ao proposto

nesta dissertação. Os quadros a seguir fornecem um resumo referente à quantidade de trabalhos identificados:

Quadro 3-3 Resultados para a busca realizada no idioma português no Google

Resultados para pesquisa em Português – Google
"Anotação Semântica de Serviços Web Restful " → 2 resultados
"Aplicação para anotação de serviços web semânticos restful" →0 resultado
"Ferramentas para anotação semântica de serviços web " → 0
"Serviços para anotação semântica de serviços web" → 0

Quadro 3-4 Resultados para a busca realizada no idioma inglês no Google

Resultado para pesquisa em Inglês – Google
"Semantic Annotation for Web Services restful" → 0 resultado
"Application for semantic annotation of web services restful" → 0 resultado
"Tools for semantic annotation of web services" → 3 resultados
"Services for semantic annotation of web services" → 0 resultado

Quadro 3-5 Resultados para a busca realizada no idioma português no Google Acadêmico

Resultado para pesquisa em Português - Google Acadêmico
"Anotação Semântica de Serviços Web Restful " → 2 resultados
"Aplicação para anotação semântica de serviços web restful" →202
" Ferramentas para anotação semântica de serviços web restful " → 0 resultado
"Serviços para anotação semântica de serviços web" → 0 resultado

Quadro 3-6 Resultados para a busca realizada no idioma inglês no Google Acadêmico

Resultado para pesquisa em Inglês - Google Acadêmico
"Semantic Annotation for Web Services restful" → 0 resultado
"Application for semantic annotation of web services" → 1 resultado
"Tools for semantic annotation of web services" → 2 resultados
"Services for semantic annotation of web services" → 0 resultado

As pesquisas realizadas na biblioteca digital da *ACM* e *IEEE* foram realizadas utilizando somente os termos em inglês. Segue abaixo resultados:

Quadro 3-7 Resultados para a busca realizada no idioma inglês na ACM Digital Library

Resultado para pesquisa em Inglês - ACM Digital Library
"Semantic Annotation for Web Services restful" → 5 resultados
"Application for semantic annotation of web services" → 1 resultado
"Tools for semantic annotation of web services" → 5 resultados
"Services for semantic annotation of web services" → 0 resultado

Quadro 3-8 Resultados para a busca realizada no idioma inglês na IEEE Digital Library

Resultado para pesquisa em Inglês - IEEE Digital Library
"Semantic Annotation for Web Services restful" → 6 resultados
"Application for semantic annotation of web services" → 140 resultados
"Tools for semantic annotation of web services" → 60 resultados
"Services for semantic annotation of web services" → 0 resultado

3.1.4 Critérios para seleção dos resultados

Os seguintes critérios de seleção foram adotados na pesquisa: (i) trabalhos que tratem primária ou secundariamente acerca de: Web Semântica, Serviços Web semânticos *restful*, anotações de Serviços Web semânticos; (ii) trabalhos que apresentem primária ou secundariamente descrição sintática e semânticas de Serviços Web, desenvolvimento de Serviços Web semânticos; (iii) trabalhos que apresentem primária ou secundariamente ferramentas para anotação de Serviços Web semânticos e que são utilizados em projetos; trabalhos que implementem em sua arquitetura o conceito de serviços.

Foram excluídos da análise do título, palavras-chave, resumo e conclusão, os seguintes estudos: (i) Estudos que não estejam disponíveis livremente para consulta na web ou Portal da Capes; (ii) Estudos que não respondam nenhuma das questões de pesquisa; (iii) Estudos Repetidos ou duplicados;

O resultado da pesquisa permitiu a seleção dos trabalhos abordados na Seção 3.2.

3.2 ANOTAÇÃO SEMÂNTICA EM SERVIÇOS WEB

Nesta seção são discutidas as ferramentas utilizadas para realizar a anotação semântica de Serviços Web e que se destinam a soluções semelhantes à apresentada neste trabalho. São consideradas propostas de ferramentas criadas para funcionar em ambiente web bem como soluções capazes de funcionar em ambientes de desenvolvimento integrado, conhecidos como IDE.

O uso da Internet pelas organizações como meio de realizar seus processos de negócios tem possibilitado o crescimento dos Serviços Web, pois facilita as integrações de serviços inter-organizacionais que acarreta em aumento de competitividade e uma maior aproximação com potenciais clientes. Dessa forma, é importante uma solução para seleção e integração de Serviços Web em tempo de execução que satisfaça às necessidades dos usuários. Nesta seção são discutidas as ferramentas existentes para a implementação de Serviços Web semânticos *restful* que são baseados no protocolo HTTP bem como as ferramentas que fornecem suporte para a criação de Serviços Web semânticos baseados no padrão SOAP.

Em cada trabalho discutido nesta seção, os autores apresentaram soluções distintas para a criação dos Serviços Web semânticos. Com o objetivo de delinear o que será abordado em cada trabalho, os seguintes critérios foram selecionados:

a) O ambiente ao qual a ferramenta irá funcionar, com o intuito de verificar qual será a sua disponibilidade para os usuários. Uma ferramenta na web poderá está disponível para milhares de usuários enquanto uma ferramenta *standalone*[54]*ou desktop* alcançaria um número de usuários menor*;*
b) Requisitos de interface homem-máquina, com o intuito de verificar características de utilidade, usabilidade e efetividade dessas ferramentas;
c) A arquitetura da ferramenta, com o intuito de verificar o quanto é possível incluir novas funcionalidades com o mínimo de alterações em sua estrutura;
d) A utilização de tecnologias que são consideradas padrões pela indústria de *software* e recomendações do W3C, com o intuito de verificar os aspectos tecnológicos que implementam as funcionalidades fornecidas pela ferramenta.

3.2.1 Uma aplicação para anotação de serviços web semânticos

Neste trabalho, (BUDINOSKI et al, 2010) propuseram como ferramenta uma aplicação web que usa ontologias com o intuito de anotar semanticamente os Serviços Web. O uso de ontologias segundo os autores possibilita que a requisição, descoberta e composição desses serviços ocorram de forma automática. A aplicação utiliza a especificação SAWSDL para fornecer a semântica em documentos WSDL.

O principal objetivo da ferramenta é fornecer uma interface flexível e de fácil uso pelo usuário, apresentando a descrição do *Web Service* em uma estrutura de árvore e permitindo a inclusão de ontologias que são utilizadas para realizar a anotação semântica. Todos os Serviços Web são armazenados em um repositório público central, que poderá ser utilizado posteriormente para a descoberta, requisição e composição desses serviços de forma automática.

As linguagens SAWSDL e WSDL utilizados são recomendações do W3C. A utilização de padrões que são recomendações do W3C torna a ferramenta com maior possibilidade de ser adotada por desenvolvedores de *software*.

A aplicação web conforme exibida na Figura 3-1 possui uma interface que contém a seguinte estrutura:

[54] Uma ferramenta ou programa é definido como *standalone* quando não necessita de um outro programa auxiliar para serem executados.

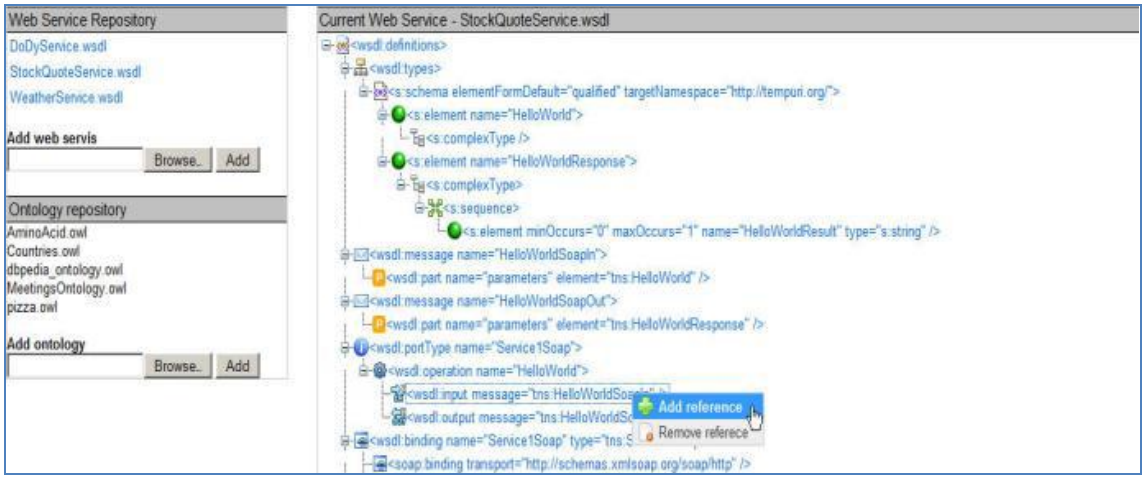
a) *Web Service Repository*

Exibe os Serviços Web que estão anotados semanticamente. Além disso, a ferramenta possibilita adicionar outros Serviços Web que poderão ser, posteriormente, anotados semanticamente.

b) *Ontology Repository*

Exibe todas as ontologias que são utilizadas para processar as anotações semânticas de um documento WSDL. Além disso, os usuários têm a opção de adicionar novas ontologias.

Figura 3-1 Aplicação Web para anotação semântica de Serviços Web



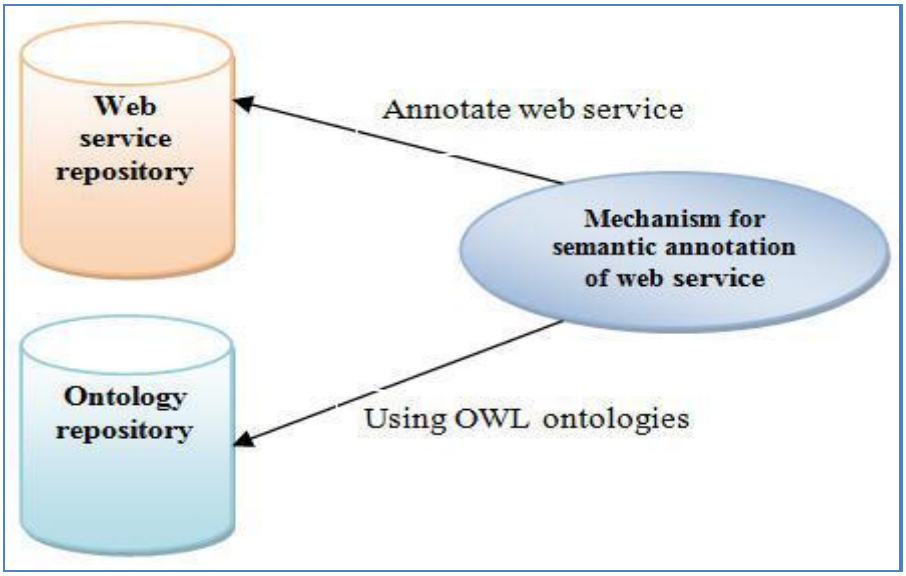
b) Fonte: Budinoski et al (2010).

a) Current Web Service

Ao selecionar um Serviço Web a partir do componente *Web Service Repository*, o Serviço Web é exibido em uma estrutura de árvore, com cada um dos seus elementos e atributos e a partir deste cenário é possível adicionar e até mesmo remover as referências semânticas.

O mecanismo utilizado para realizar a anotação semântica de Serviços Web é ilustrado na Figura 3-2. Ao fazer uso deste mecanismo de anotação semântica de Serviços Web e armazenando esses Serviços Web em um repositório público central a aplicação possibilita implementar os processos de descobrir, invocar e compor serviços de forma automática.

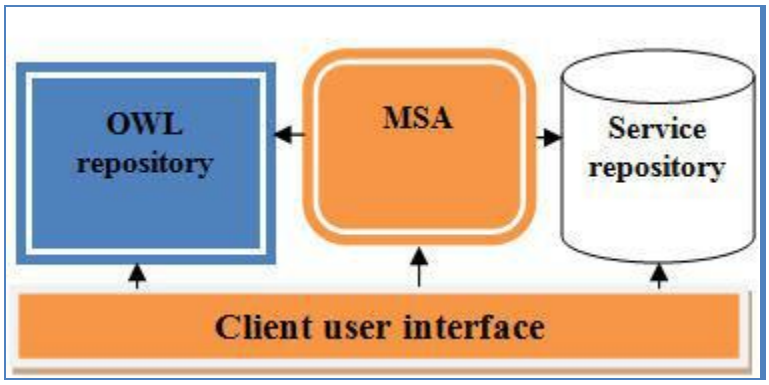
Figura 3-2 Mecanismo para anotação de Serviços Web semânticos



Fonte: Budinoski et al (2010).

A arquitetura deste projeto é apresentada na Figura 3-3. Os principais componentes dessa arquitetura incluem: (i) um repositório *OWL Ontology*; (ii), um repositório de serviços *(Service Repository)*; (iii) um mecanismo para a anotação semântica *(MSA*) e, (iv) uma interface de usuário (*Client user interface*), os quais são discutidos a seguir:

Figura 3-3 Arquitetura da aplicação



Fonte: Budinoski et al (2010).

a) (*OWL repository*, repositório de ontologias OWL*)*

Este repositório mantém o controle de todas as ontologias que são usadas para processar as anotações de um documento WSDL e/ou XML *Schema Defintion*. Os elementos desses documentos serão anotados utilizando os conceitos semânticos a partir das ontologias armazenadas neste repositório. O repositório é extensível, isto é, os usuários possuem a opção de adicionar novas ontologias.

b) *Service Repository* (Repositório de Serviços Web*)*

Neste repositório é mantido o controle de todos os Serviços Web anotados semanticamente. Os Serviços Web são representados pelos documentos WSDL. O repositório de Serviços Web é extensível, e os usuários têm a opção para adicionar novos serviços web. Posteriormente este repositório poder ser usado para a descoberta, invocação e composição automática de serviços web.

c) MSA (Mechanism for semantic annotation, Mecanismo para anotação semântica)

Este componente fornece a interoperabilidade necessária entre o repositório de ontologia OWL e o repositório de Serviços Web. O MSA trabalha como um mediador entre esses dois repositórios e possui como principal objetivo fornecer mecanismos pelos quais os conceitos determinados a partir dos modelos semânticos podem ser referenciados aos elementos dos documentos WSDL. Conforme já citado, o padrão SAWSDL é utilizado para anotar semanticamente os elementos de um documento WSDL.

d) Client user interface (Interface com o usuário)

Fornece o suporte de interação do usuário com a aplicação web. Todos os Serviços Web são representados como uma estrutura de árvore, facilitando a utilização da aplicação. O vocabulário semântico determinado pelas ontologias OWL podem ser adicionados ou removidos usando o menu de contexto que pode ser ativado através de um clique com o botão direito nos elementos do documento WSDL.

Apesar das vantagens apresentadas pelos autores, a ferramenta proposta não fornece as seguintes funcionalidades:

a) anotação de Serviços Web *restful* semânticos;

b) utilização de ontologias existentes na Internet;

c) acessar Serviços Web que estejam na Internet para realizar a anotação semântica;

d) não disponibiliza serviços para acesso as funcionalidades oferecidas pela aplicação.

3.2.2 SWEET

Com o objetivo de possibilitar aos desenvolvedores de *software* fornecer semântica aos Serviços Web *restful* que possuem suas descrições sintáticas disponibilizados em documentos HTML, Maleshkova, Pedrinaci e Domingue (2009) criaram a ferramenta SWEET.

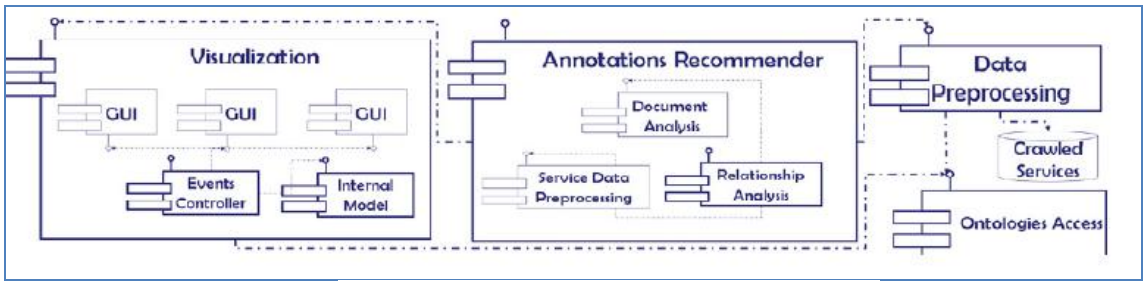
SWEET é um acrônimo para *Semantic Web Services Editing Tool.* Trata-se de uma aplicação que é iniciada no navegador web e construída utilizando as tecnologias *JavaScript*[55] e ExtGWT[56] tem como principal objetivo a anotação semântica de Serviços Web *restful*.

A interface do SWEET carrega em sua página inicial a descrição do Serviço Web *restful* e oferece funcionalidades que possibilitam aos usuários anotar as propriedades e semânticas referentes a esses serviços.

A arquitetura do SWEET, ilustrada na Figura 3-4, é composta de três componentes:

a) Visualization
b) Data Preprocessing
c) Annotations Recommender

Figura 3-4 Arquitetura SWEET



Fonte: Maleshkova, Pedrinaci e Domingue (2009).

O componente *Visualization*, ilustrado na Figura 3-5, possui três componentes principais:

a) Semantic Description

Essa estrutura é responsável em exibir o estado atual da anotação semântica do Web Service restful.

b) Navigator

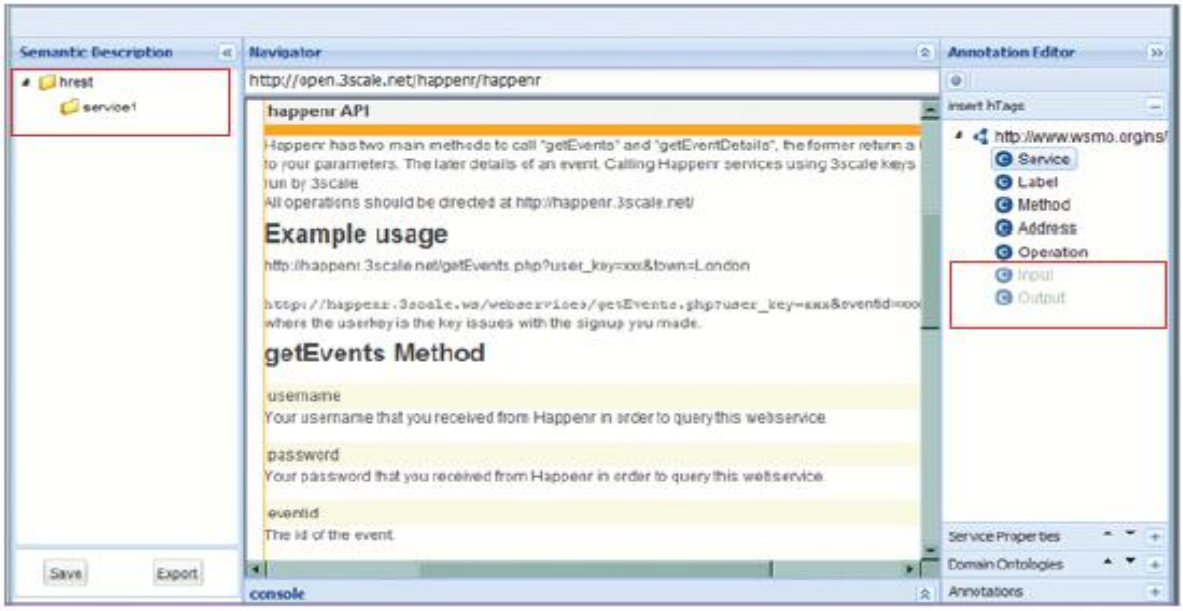
Nessa estrutura é carregado o documento HTML que contém a descrição sintática dos Serviços Web restful.

c) Annotation Editor

[55] http://www.w3schools.com/js/
[56] http://extjs.com/products/gxt

Essa estrutura disponibiliza funcionalidades capazes de manipular o DOM (*Document Object Model*)[57] do documento HTML que contém a descrição sintática dos Serviços Web *restful*.

Figura 3-5 Componente de Visualização SWEET



Fonte: Maleshkova, Pedrinaci e Domingue (2009).

O componente *Annotations Recommender* age de forma a auxiliar o usuário a anotar semanticamente um *Web Service*, pois realiza sugestões de anotações apropriadas para o serviço como um todo e de propriedades individuais. Esse componente utiliza uma abordagem de recomendação híbrida, pois o conteúdo da recomendação pode ser implementado por anotações realizadas anteriormente em serviços similares e também pode ser baseada em uma recomendação ontológica.

O componente *Data Preprocessing* tem como principal objetivo preparar os dados para o componente de visualização além de possuir um mecanismo de *cache*.

Nesse trabalho, os autores utilizam os micros formatos hRests (*HTML for RESTfull Services*)[58] e MicroWSMO como soluções para fornecer a representação semântica dos Serviços Web *restful* e apresentam três benefícios na adoção dessas tecnologias:

a) o uso de microformatos permite que descrições de Serviços Web *restful* em documentos HTML possam ser distinguidos de HTML simples;

[57] http://www.w3.org/DOM/
[58] http://iserve.kmi.open.ac.uk/ns/hrests/hrests-2013-10-03.html

b) o próprio serviço de busca pode ser melhorado e automatizado usando a informação semântica que está anexada no documento HTML;

c) MicroWSMO fornece anotações semânticas de Serviços Web *restful* de modo similiar ao conceito utilizado pela SAWSDL e baseia-se na ontologia WSMO-Lite[60] que é uma especificação para a anotação de Serviços Web. De acordo com os autores tanto MicroWSMO como SAWSDL podem utilizar a ontologia WSMO-LITE, dessa forma Serviços Web *restful* com descrição em HTML e Serviços Web descritos em WSDL 1.1 podem ser recuperados utilizando as mesmas consultas.

A Listagem 3-1 ilustra um exemplo para recuperar Serviços Web com descrições sintáticas em documentos WSDL e contendo anotações semânticas em linguagem SAWSDL e Serviços Web *restful* contendo anotações semânticas no microformato MicroWSMO. A consulta semântica, nesse exemplo, recupera todos os Serviços Web com o nome "*Username*".

Listagem 3-1 Exemplo de consulta um serviço

```
1 SELECT DISTINCT?s
2 WHERE {
3 ?s rdf : type wsl : Service .
4 ?s sawsdl:modelReference <http://example.com/data/onto.owl#Username>
5 }
```

Nesse trabalho não foram identificadas as seguintes funcionalidades:

a) os Serviços Web *restful* descritos sintaticamente em WSDL não são anotados semanticamente;

b) utilização de ontologias que estejam disponíveis na Internet;

c) acesso a Serviços Web que estejam na Internet para realizar a anotação semântica;

d) estabelecimento de um repositório com o intuito de promover a automação nas tarefas de requisição, descoberta e composição dos Serviços Web *restful*;

e) disponibilização de serviços para acesso as funcionalidades oferecidas pela ferramenta.

[60] http://www.wsmo.org/ns/wsmo-lite/

3.2.3 WSMO Studio

Dimitrov et al (2007), apresentam um protótipo com três objetivos principais:

a) Fornecer um modelo que apoie e desenvolva Serviços Web semânticos utilizando a especificação WSMO (*Web Service Modeling Ontology*)[61], tornando a tecnologia mais simples de ser utilizada e que facilite a adoção dos usuários;
b) Proporcionar um ambiente integrado que capacite o aumento da produtividade do usuário;
c) Proporcionar um ambiente extensível onde seja possível adicionar novas funcionalidades ou alterar uma funcionalidade existente.

O WSMO Studio (*Web Service Modelling Ontology Studio*) é um protótipo open source construído sob a IDE ECLIPSE[62] e possui um ambiente integrado de modelagem e construção de Serviços Web semânticos.

O WSMO Studio oferece uma solução que permite ir além da anotação semântica de Serviços Web semânticos e dessa forma suporta um maior número de funcionalidades:

a) Modelagem de ontologias;
b) Anotação semântica de Serviços Web;
c) Trabalha com repositórios semânticos para a publicação, navegação e consulta de ontologias WSMO, serviços, metas e mediadores;
d) Descoberta de serviços baseado em objetivos;
e) Especificação de composições de serviços (ou seja, orquestração e coreografia
f) Interfaces);
g) Interação com ambientes de execução de serviços da Web Semântica tais como WSMX3[63] e IRS-III[64].

Com o intuito de atender aos objetivos e funcionalidades o WSMO Studio possui as seguintes camadas:

a) *Core Components*

---

[61] http://www.w3.org/Submission/WSMO/

[62] ECLIPSE é uma IDE que fornece suporte ao desenvolvimento de aplicações escritas em linguagem de programação Java.

[63] http://www.wsmx.org

[64] http://kmi.open.ac.uk/projects/irs/

Fornece funcionalidades comuns a todas as outras camadas do WSMO Studio.

b) *WSMO editor*

Fornece a interface de usuário e possui como principal objetivo a modelagem de ontologias WSMO, metas e serviços.

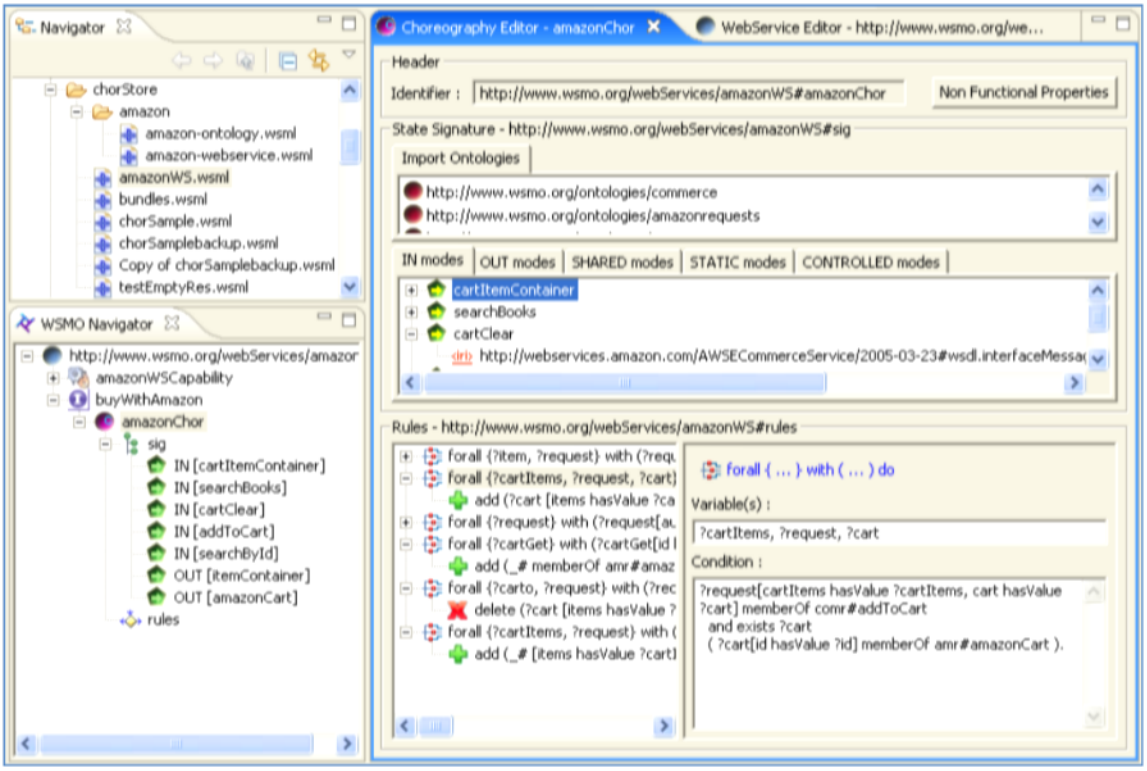
c) *Choreography editor*

O *Choreography Editor*, ilustrado na Figura 3-6, fornece uma interface para descrever a coreografia WSMO. As descrições das coreografias criadas no WSMO Studio são utilizadas em ambientes de execução como WSMX durante a execução dos Serviços Web semânticos.

d) *SAWSDL editor*

O editor SAWSDL, ilustrado na Figura 3-7, fornece a funcionalidade para adicionar anotações semânticas em documentos WSDL de acordo com a especificação SAWSDL sendo utilizado em conjunto com WSMO. A interface do usuário permite incluir anotações semânticas aos seguintes elementos: (i) tipos simples e complexos, (ii) mensagens, (iii) operações e (iv) interfaces.

Figura 3-6 *Choreography Editor*

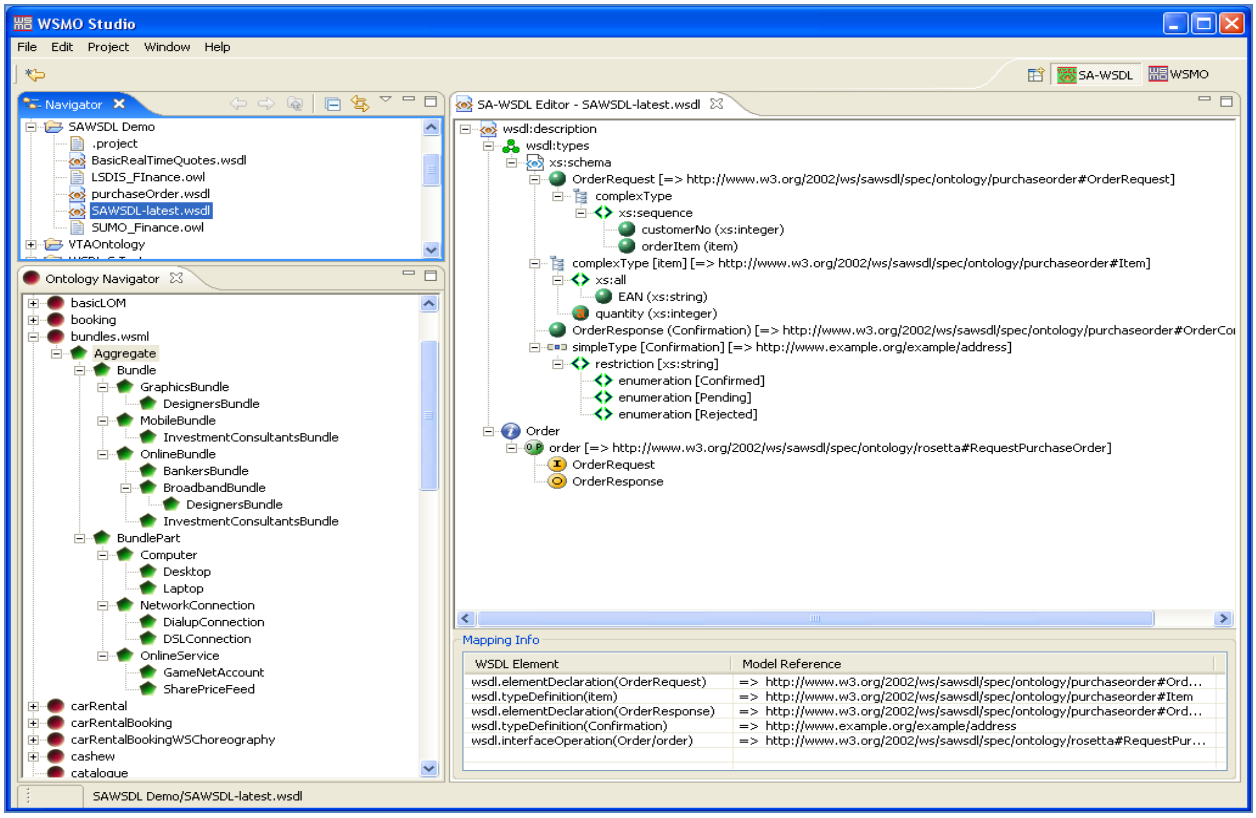


Fonte: Dimitrov et al (2007).

O editor SAWSDL (Figura 3-7) não fornece suporte ao mapeamento que pode ser realizado entre os dados semânticos e elementos do esquema XML o que inviabiliza, por

exemplo, realizar a transformação da estrutura de dados indicada no documento WSDL em um formato determinado pelo modelo semântico. A especificação SAWSDL fornece suporte a esse mapeamento através dos atributos de extensão *liftingSchemaMapping* e *loweringSchemaMapping* explicados na Seção 2.7.

Figura 3-7 SAWSDL editor



Fonte: Dimitrov et al (2007).

e) *Repository front-end*

O *Repository front-end* fornece uma abstração de um repositório de serviços da Web Semântica que possibilita armazenar e consultar descrições WSMO.

f) *Service discovery front-end*

A descoberta de serviços WSMO baseia-se em descoberta baseada em metas, ou seja, de acordo com os autores o usuário fornece uma solicitação representado como um objetivo WSMO e com algumas informações adicionais, como restrições ou custo de serviço de qualidade, e o componente de descoberta retorna uma lista de serviços classificados de acordo com os critérios de classificação.

g) ***Integrated validator and reasoners***

O WSMO Studio fornece integração com WSML-Flight [65] e WSML-DL [66] via framework *WSML2Reasoner* [67]. Esses validadores são utilizados para validação e testes aos construir ontologias WSMO.

Com o WSMO Studio objetiva-se realizar a modelagem de Serviços Web semânticos e a ferramenta já está sendo utilizado em dois projetos de pesquisa para atingir esse objetivo: (i) SUPER[68] e o (ii) *Semantic GOV* [69].

Na análise desse trabalho identificou-se como vantagens: (i) a possibilidade de modelar ontologias; (ii) anotação semântica de Serviços Web*;* (iii) descoberta de serviços baseado em objetivos; (iv) possibilidade de trabalhar com repositório semânticos para atendimento às tarefas de publicação, navegação e consulta e (v) especificação de composição de serviços, ou seja, suporte a orquestração e coreografia de Serviços Web.

Além das mesmas questões identificadas no projeto SWEET, a ferramenta WSMO Studio apresenta as seguintes desvantagens:
(i) os autores trabalharam com a anotação semântica de documentos WSDL 1.0 seguindo apenas uma parte da especificação SAWSDL, pois não fornece suporte ao mapeamento de esquema. Oliveira (2009) afirma que os atributos de extensão (*liftingSchemaMapping* e *loweringSchemaMapping*) fornecidos pelo mapeamento de esquema são utilizados para realizar ajustes semânticos nas mensagens trocadas entre os Serviços Web. Esses ajustes possuem o objetivo de transformar a estrutura de dados especificada no documento WSDL para o formato exigido pelo modelo semântico associado e vice-versa; (ii) outra desvantagem dessa ferramenta é a não utilização de ontologias definidas em linguagem OWL. De acordo com (PANZIERA et al 2010), a OWL é a linguagem mais utilizada para a construção de ontologias com o objetivo de determinar a descrição semântica na web.

3.2.4 RadiantWeb

Proposta por Guttula (2012) a *RadiantWeb* é uma ferramenta baseada na Web que oferece suporte para anotações de Serviços Web dispostos na arquitetura REST e SOAP. A

[65] http://tools.sti-innsbruck.at/wsml2reasoner/

[66] http://tools.sti-innsbruck.at/wsml2reasoner/

[67] http://tools.sti-innsbruck.at/wsml2reasoner/

[68] http://www.ip-super.org/

[69] http://semantic-gov.org/

ferramenta favorece a anotação de serviços escritos sintaticamente em WSDL 1.1 e WADL. A ferramenta emprega mecanismos de recomendação[70] reduzindo o esforço do usuário na seleção das ontologias. O autor destaca dois problemas na realização da anotação semântica de Serviços Web: (i) localizar a ontologia apropriada, (ii) e após localização, identificar os conceitos relevantes dessa ontologia. Segundo o autor, esses problemas aumentam o número de horas para realizar a anotação semântica um *Web Service* e a utilização de recomendação pode acelerar consideravelmente o processo de anotação.

Para localizar a ontologia apropriada, a ferramenta utiliza o recomendador NCBO (*National Center for Biomedical Ontology*)[71]. Esse recomendador recebe como entrada de dados uma *string* e oferece como resposta uma lista ordenada das ontologias que poderão ser utilizadas para realizar a anotação semântica do *Web Service*. Esse recomendador sugere as ontologias que estão registradas nos repositórios NCBO BioPortal e no UMLS (*Unified Medical Language System*)[72].

Duas maneiras de recomendar os conceitos relevantes da ontologia selecionada são utilizadas: (i) fornecendo a melhor recomendação para cada elemento do documento de descrição do *Web Service*, (ii) fornecendo *N* recomendações para um único elemento do documento de descrição do *Web Service*.

Dessa forma, o mecanismo de recomendação é responsável por sugerir os conceitos mais prováveis para anotações, realizando a adequação dos conceitos ontológicos para os elementos de documentos WSDL / WADL.

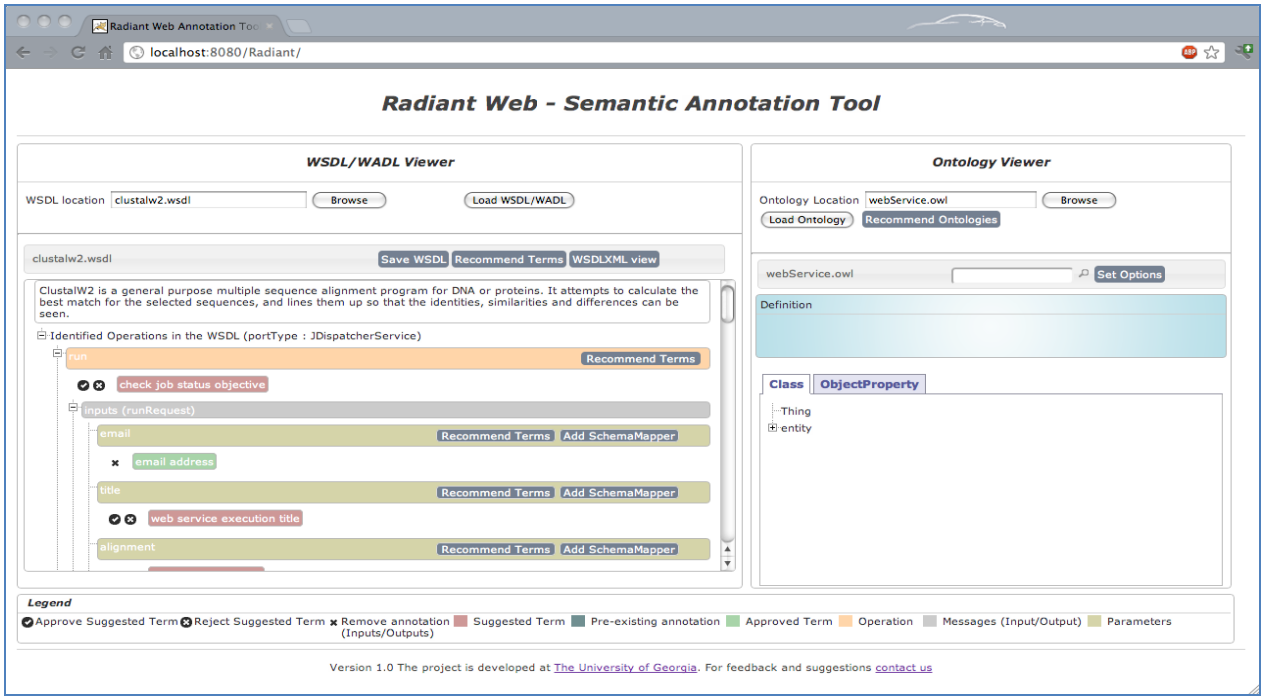
---

[70] http://www.inf.unibz.it/~ricci/papers/intro-rec-sys-handbook.pdf

[71] http://nar.oxfordjournals.org/content/39/suppl_2/W541.short

[72] http://nar.oxfordjournals.org/content/32/suppl_1/D267.full

Figura 3-8 Interface da Ferramenta *RadiantWeb*



Fonte: Guttula (2012).

A ferramenta *RadiantWeb* possui uma interface gráfica composta por dois painéis principais:

a) WSDL/WADL *Viewer*

O WSDL/WADL *Viewer* possui a responsabilidade de carregar e exibir um documento que contém a descrição da *Web Service*.

b) Ontology Viewer

O *Ontology Viewer* possui a responsabilidade de carregar e suportar a navegação e consulta de ontologias.

A arquitetura do sistema segue o padrão MVC (*Model View Controller*) [73]. O início do fluxo de dados é determinado pela seleção do documento que necessita da anotação semântica. Esse documento pode está localizado no sistema de arquivos local ou em algum endereço da Web. Uma requisição feita em AJAX (*Asynchronouus Java Script and XML*)[74] é

[73] http://msdn.microsoft.com/en-us/library/ff649643.aspx

[74] http://www.w3schools.com/ajax/default.ASP

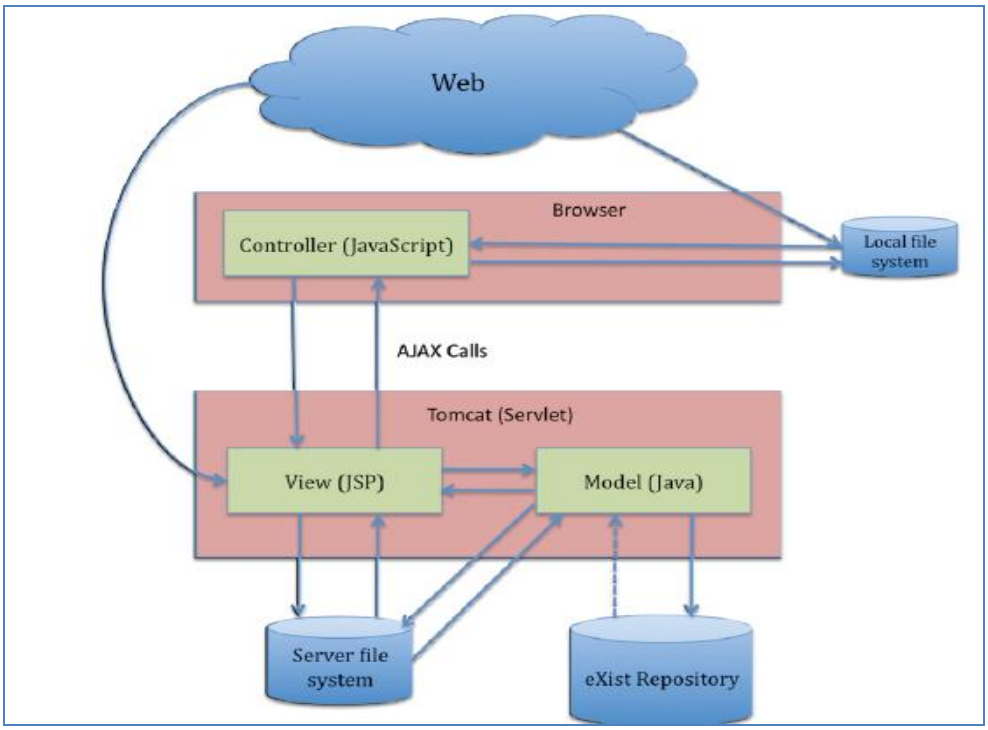
realizada pela camada controladora (*Controller*) para a camada Visão (*View*), com o objetivo de armazenar o documento no sistema de arquivos do servidor. Uma vez que o documento é salvo, ele é carregado através da camada Modelo (*Model*) que envia os dados formatados pela camada Visão à camada Controlador.

O controlador apresenta o documento de descrição sintática do *Web Service* através de duas formas:

a) em uma estrutura de árvore listando as operações do serviço, bem como suas entradas e saídas;
b) em um documento XML.

O fluxo dessa comunicação é exibido na Figura 3-9.

Figura 3-9 Arquitetura da *RadiantWeb*



Fonte: Guttula (2012).

Na análise desse trabalho, identificou-se como vantagens: (i) utilização de algoritmos de recomendação para sugerir as ontologias mais prováveis para a anotação semântica dos Serviços Web, (ii) a ferramenta é uma aplicação web, o que pode tornar a aplicação disponível para milhares de usuários. Como desvantagens, a ferramenta não suporta Serviços Web descritos utilizando a versão WSDL 2.0 que é uma recomendação do W3C para

descrição de Serviços Web *restful*, sendo assim é incapaz de anotar os serviços *restful* descritos nessa versão da linguagem WSDL.

3.2.5 Iridescent: uma ferramenta para descrição rápida de serviços web semânticos

Stavropoulos, Vrakas e Vlahavas (2011) propõem uma ferramenta para a criação e edição de Serviços Web semânticos, chamada *Iridescent*, baseado nas especificações OWL, WSDL e SAWSDL. Essa ferramenta, assim como a *RadiantWeb*, oferece mecanismos de recomendação que auxiliam na pesquisa das ontologias mais apropriadas para a anotação semântica dos Serviços Web.

Os autores deste trabalho não apresentaram o projeto de arquitetura da ferramenta *Iridescent*, entretanto, apresentaram um conjunto de requisitos que são atendidos e certamente influenciaram na elaboração da sua arquitetura. Os principais requisitos estão listados Quadro 3-9:

Quadro 3-9 Requisitos ferramenta *Irisdescent*

Compatibilidade
A ferramenta é independente de plataforma, pois tem como objetivo atingir o maior público possível. Para atender a esse requisito, os autores optaram em utilizar a plataforma Java.
Disponibilidade
Os recursos da ferramenta devem estar sempre disponíveis. Segundo os autores, em uma solução online a disponibilidade é limitada por restrições de conectividade. Por essa razão, a ferramenta foi desenvolvida como uma aplicação *desktop* não exigindo um processo de instalação. As ontologias e descrições dos serviços podem ser carregadas tanto a partir de um disco local como da Internet.
Usabilidade
A ferramenta deve ser fácil de utilizar mesmo para usuários com pouco conhecimento em Web Semântica.
Extensibilidade
A ferramenta deve ser extensível para futuras tecnologias e formatos de desenvolvimento de *software*. Atualmente a ferramenta suporta ontologias descritas em OWL / RDF, mas pode ser estendida para atender a outras linguagens.

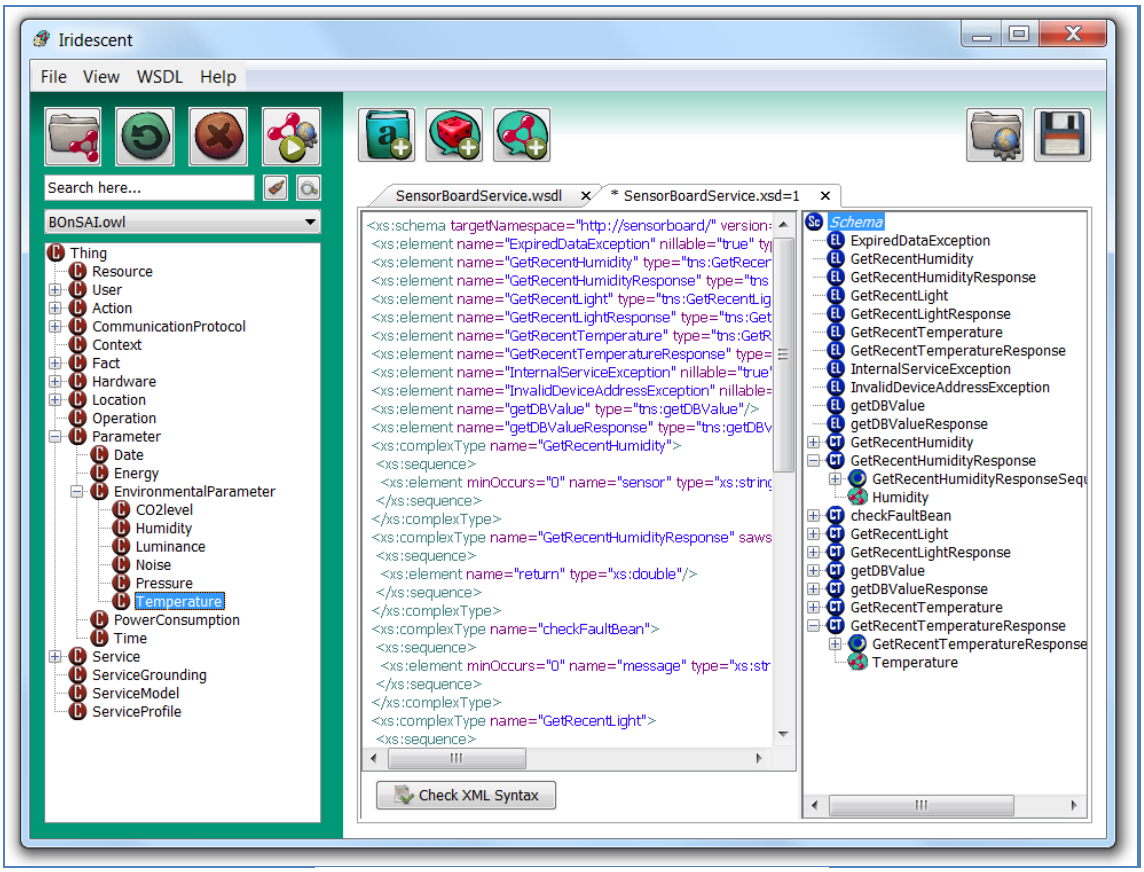
Fonte: Baseado em Stavropoulos, Vrakas e Vlahavas (2011).

A Figura 3-10 ilustra a tela principal da ferramenta *Iridescent* que é composta por três painéis principais nos quais cada conceito é destacado em uma cor diferente:

a) Em vermelho estão as ontologias definidas em OWL;
b) Em verde estão a descrições dos serviços definidos em WSDL;
c) Em azul estão os elementos semânticos.

Através da utilização da ferramenta *Iridescent*, os Serviços Web *restfful* poderão ser anotados semanticamente, pois a ferramenta suporta o uso de documentos descritos em WSDL 2.0.

Figura 3-10 Tela principal da ferramenta *Iridescent*



Fonte: Stavropoulos, Vrakas e Vlahavas (2011).

Na análise desse trabalho identificou-se como vantagens: (i) utilização de algoritmos de recomendação para sugerir as ontologias mais prováveis para a anotação semântica dos Serviços Web, (ii) anotação semântica de Serviços Web *restful* que possuem descrição sintática em WSDL. Todavia, os requisitos indicados no trabalho dos autores não contemplaram a implementação de um repositório para as ontologias e Serviços Web semânticos, nem tampouco sua disponibilização na Internet. A ferramenta também não disponibiliza as suas funcionalidades através de serviços.

3.3 CONSIDERAÇÕES FINAIS

Nesta seção são feitas considerações finais acerca dos trabalhos correlatos discutidos neste capítulo. Os resultados foram resumidos em um quadro comparativo (Quadro 3 - 12) com os requisitos estabelecidos no (Quadro 3 - 11).

Budinoski et al (2010) defendem o uso de uma aplicação web para realizar a anotação semântica de Serviços Web no padrão SOAP. Fornece uma arquitetura que possibilita estabelecer um repositório tanto para os Serviços Web como para as Ontologias construídas em linguagem OWL.

A aplicação SWEET possui como principal objetivo a anotação semântica de Serviços Web *restful*. Essa aplicação utiliza hRESTS e MicroWSMO em sua arquitetura para prover a solução.

O WSMO Studio fornece uma perspectiva única para modelagem de Serviços Web semânticos e uma das tarefas atendidas pela ferramenta é a anotação semântica de Serviços Web. Essa ferramenta realiza anotações semânticas em documentos WSDL 1.0 de acordo com a uma parte da especificação SAWSDL.

A ferramenta RADIANTWEB realiza a anotação semântica de Serviços Web descritos sintaticamente em WSDL e WADL. A RADIANTWEB fornece vários recursos incluindo o suporte a pesquisa de ontologias na Internet além de trabalhar com conceitos de algoritmos de recomendação. A aplicação não suporta WSDL na versão 2.0. Os conceitos ontológicos definidos por essa aplicação são realizados através de OWL.

A aplicação IRIDESCENT realiza a anotação de Serviços Web descritos em documentos WSDL 1.0 e WSDL 2.0. É uma ferramenta *desktop* que possui como requisito a capacidade de ser independente de plataforma, pois foi codificada em linguagem de programação Java. Utiliza a especificação SAWSDL para realizar a anotação semântica dos Serviços Web. Um resumo das principais propostas de cada um dos trabalhos estudados é descrito no Quadro 3-10.

Quadro 3-10 Resumo dos trabalhos relacionados

TRABALHOS	PROPOSTA
Aplicação web para anotação de serviços web semânticos	Utilizar o mecanismo SAWSDL em conjunto com o WSDL para anotar Serviços Web semânticos.
SWEET	Descrever Serviços Web *restful* recebendo como entrada uma página da Web em HTML que descreve a API Web e oferece funcionalidades que permitem aos usuários anotar semântica as propriedades do serviço.
WSMO Studio	*Plugin* para a IDE eclipse que possibilita modelar e construir Serviços Web semânticos utilizando abordagem WSMO.
RADIANTWEB	Uma ferramenta para anotar Serviços Web *restful* e SOAP. A solução utiliza algoritmos de recomendação com o objetivo de reduzir o tempo do usuário na tarefa de anotar semanticamente os Serviços Web. Não suporta WSDL 2.0. Os conceitos ontológicos são determinados por OWL.
IRIDESCENT	Ferramenta *Desktop* construída em Java que utiliza o mecanismo SAWSDL para anotação de serviços web. Suporta as recomendações WSDL 1.0 e 2.0. Assim como a ferramenta *RadiantWeb* utiliza algoritmos de recomendação para auxílio na seleção das ontologias.

As ferramentas supracitadas apresentam soluções distintas. Algumas utilizam tecnologias que são recomendações do W3C, a exemplo das especificações WSDL e SAWSDL, outras utilizam abordagens que são empregadas na criação de Serviços Web *restful*, porém não são recomendações do W3C.

Com base nas propostas estudadas, são elencados a seguir alguns requisitos que são considerados na implementação da ferramenta proposta nesta dissertação. Esses requisitos são utilizados também como critério de comparação entre os trabalhos correlatos. Outro fator que influenciou na definição dos requisitos foi a capacidade de abstrair detalhes específicos de cada tecnologia utilizada na anotação semântica de Serviços Web. Esses requisitos propostos estão especificados no Quadro 3-11 e justificados no Quadro 3-12.

Quadro 3-11 Requisitos para comparação dos trabalhos relacionados

Requisitos (Código)	Requisitos (Descrição)
R1	Realizar a anotação semântica de Serviços Web *restful* descritos com WSDL 2.0
R2	Realizar a anotação semântica de serviços web padrão REST
R3	Permitir a utilização de Ontologias disponíveis na web
R4	Utilizar tecnologias que são recomendações do W3C
R5	Utilizar conceitos ontológicos definidos em OWL
R6	Manter um repositório de Serviços Web Semânticos
R7	Manter um repositório de Ontologias
R8	Ser extensível (Utilização de Serviços)
R9	Realizar a anotação de serviços no padrão SOAP

Os requisitos identificados são justificados no Quadro 3-12:

Quadro 3-12 Justificativa para seleção dos requisitos

R1	Durante a análise das soluções encontradas para anotação semântica de Serviços Web *restful* foram identificadas ferramentas que realizam a descrição sintática desses serviços através de diferentes especificações como, por exemplo, SA-REST e WADL, entretanto, Mandel (2008) afirma que a especificação WSDL 2.0 foi projetada para descrever Serviços Web *restful*, sendo uma recomendação do W3C desde junho de 2007;
R2	Aplicações da Web 2.0 são implementadas conforme o estilo arquitetural REST. Ferreira (2011) afirma que as pesquisas em Serviços Web Semânticos têm sido maiores na arquitetura orientada a serviço que são baseadas em SOAP, sendo pouco as soluções que propõe fornecer semântica a serviços implementados conforme o estilo arquitetural REST;
R3	Durante a análise dos trabalhos correlatos foi verificado quais ferramentas realizam uso de ontologias OWL pré-existentes na web e foi verificado que a maioria das ferramentas analisadas não implementam essa funcionalidade;
R4	Utilizar recomendações W3C facilita uma suposta adoção da solução pela indústria de *software*;
R5	O vocabulário semântico determinado na solução deverá ser constituído por ontologias descritas em OWL por se tratar de uma recomendação do W3C. De acordo com (PANZIERA et al. 2012), a OWL é a linguagem mais utilizada para a construção de ontologias com o objetivo de determinar a descrição semântica na web;

R6	Armazenar os Serviços Web *semânticos* que poderão ser utilizados futuramente para realização de buscas semânticas. Hobold e Siqueira (2012) fornecem uma abordagem para descoberta de serviços com base nas informações anotadas em documentos WSDL a partir da especificação SAWSDL;
R7	Armazenar as ontologias que são utilizadas para processar as anotações semânticas de um documento WSDL e/ou XML *Schema Definition*. Os elementos desses documentos serão anotados usando os conceitos semânticos dessas ontologias que estão armazenadas no repositório ou existentes na web;
R8	A arquitetura da ferramenta deverá ser projetada de forma a possibilitar o incremento de novos requisitos mitigando o máximo possível de alterações na sua estrutura e possibilitando o seu acesso em diferentes cenários. Para isso a ferramenta deve ser projetada utilizando serviços. De acordo com Erl (2009), a disponibilização de serviços favorece ao reuso, pois estabelece as condições para que diferentes cenários possam compartilhar os mesmos recursos (e.g.: web, *desktop*, entre outros);
R9	Durante a análise dos trabalhos correlatos foram identificadas soluções que realizam a anotação semântica de Serviços Web que são construídos de acordo com o protocolo SOAP. Esses trabalhos serão estudados com o objetivo de comparar e analisar os requisitos funcionais implementados na realização da anotação semântica e contemplá-los na construção da ferramenta.

O Quadro 3-13 resume a comparação, com base nos requisitos especificados no Quadro 3-11, entre as ferramentas discutidas neste capítulo. No Quadro 3-13, ***sim*** significa que a ferramenta atende integralmente o requisito correspondente e ***não*** é o caso contrário, quando a aplicação não atende ao requisito. Quando marcado como ***parcialmente***, a ferramenta em questão não atende completamente o requisito correspondente.

Quadro 3-13 Avaliação dos trabalhos relacionados

	R1	R2	R3	R4	R5	R6	R7	R8	R9
Anotação Semântica	Não	Não	Não	Sim	Sim	Sim	Sim	Não	Sim
SWEET	Sim	Não	Não	Parcialmente	Não	Não	Não	Não	Sim
WSMO Studio	Não	Não	Não	Parcialmente	Não	Sim	Sim	Não	Sim
RADIANTWEB	Sim	Não	Sim	Parcialmente	Sim	Não	Não	Não	Sim
IRIDESCENT	Sim	Sim	Sim	Sim	Sim	Não	Não	Não	Sim

A ferramenta apresentada por (BUDINOSKI et al, 2010) não atende aos requisitos **R1** e **R2,** pois não suporta a anotação semântica de Serviços Web que seguem a abordagem REST. A ferramenta também não permite a utilização de ontologias preexistentes na internet e dessa forma não atende ao requisito **R3**. Com relação aos requisitos **R4, R5, R6, R7e R9** pode-se afirmar que a ferramenta atende. Quanto ao requisito **R8** a ferramenta não suporta o incremento de novos formatos de desenvolvimento e nem fornece acesso às suas funcionalidades por meio de serviços.

Com relação a ferramenta SWEET os autores utilizam como solução hRests e MicroWSMO para fornecer a representação semântica dos serviços que estão descritos sintaticamente em documentos HTML, entretanto, os serviços que possuem suas descrições sintáticas expostas em documentos WSDL não são anotados semanticamente e dessa forma não suporta o requisito **R1**. A principal proposta da ferramenta é a descrição de Serviços Web implementados conforme o paradigma REST o que contempla o requisito **R2.** Os requisitos **R3, R5, R6, R7 E R8** não são atendidos pela ferramenta. O requisito **R4** é atendido parcialmente, pois algumas tecnologias empregadas na solução não são recomendações do W3C, a exemplo do WSMO. A linguagem para determinar um vocabulário ontológico é a OWL.

O WSMO Studio é uma solução que integra diversas ferramentas para tratar os Serviços Web semânticos. Nesse trabalho é considerado para efeitos de comparação o SAWSDL Editor. Os requisitos **R1, R2, R3 e R5** não são atendidas pela ferramenta. A ferramenta não suporta WSDL 2.0 e dessa forma não fornece suporte aos Serviços Web *restful* sendo que o vocabulário ontológico é determinado por WSMO fazendo com que o requisito **R4** seja atendido parcialmente. A ferramenta também não atende aos requisitos **R6**, **R7** e **R8**.

A aplicação RADIANTWEB é baseada na web e cumpre grande parte dos requisitos estabelecidos. O requisito **R4** é atendido parcialmente, pois a descrição sintática dos serviços web utiliza o padrão WADL que não tornou de fato uma recomendação do W3C. A ferramenta não atende aos requisitos **R6**, **R7** e **R8**.

A solução IRIDESCENT é entre os trabalhos avaliados a aplicação mais completa ao atendimento aos requisitos determinados no Quadro 3-11. Os requisitos **R6, R7 e R8** não foram verificados sendo atendido pela aplicação.

As propostas discutidas neste capítulo foram incluídos pela proximidade com o tema e com as tecnologias utilizadas na elaboração desta dissertação. Os requisitos destacados no Quadro 3-11 são os previstos na ferramenta apresentada nesta dissertação.

O capítulo seguinte discute, devido à falta de completeza e/ou deficiência demonstradas pelas ferramentas na implementação e suporte aos requisitos listados, a solução proposta, suas características, implementações e melhorias incorporadas com base nos projetos tratados neste capítulo. Vale salientar que, até a presente data, não foram identificados trabalhos que tenham como referência uma ferramenta que realize a anotação semântica de Serviços Web através do uso de serviços.

4 SWS EDITOR: UMA FERRAMENTA BASEADA EM SERVIÇOS PARA ANOTAÇÃO SEMÂNTICA DE SERVIÇOS WEB *RESTFUL*

Este capítulo destina-se a apresentação da solução proposta para realizar a anotação semântica de Serviços Web *restful*. Para isso são discutidas as decisões de projeto e apresentada a arquitetura da solução. Este capítulo está estruturado da seguinte forma: a Seção 4.1 discute conceitualmente a solução, a Seção 4.2 exibe e descreve a arquitetura da ferramenta. A Seção 4.3 discorre sobre os serviços oferecidos pela ferramenta, no qual é representada através da utilização de diagrama de atividades e a Seção 4.4 discute os aspectos da implementação dos serviços oferecidos pela ferramenta, os quais são representados através de diagramas de classes. A arquitetura e modelagem da aplicação possibilitará um melhor entendimento da solução empregada para a construção de Serviços Web *restful* semânticos.

4.1 SWS EDITOR

Maleshkova, Pedrinaci e Domingue (2012) afirma que em razão das limitações das tecnologias atuais, a adoção de Serviços Web *restful* é prejudicada visto que esses serviços não possuem uma representação semântica de modo que um computador ou outro *Web Service* possam interpretar informações fornecidas por esses serviços. Portanto, com o objetivo de obter as vantagens proporcionadas pela anotação semântica, um maior número de Serviços Web necessita ser anotado. Gutulla (2012) discute a importância da construção de ferramentas apropriadas para auxiliar na atividade de fornecer semântica aos Serviços Web visto que a anotação sendo realizada de maneira manual por um desenvolvedor de *software* poderia custar inúmeras horas de esforço para sua conclusão.

Para realizar a anotação semântica de Serviços Web *restful* descritos sintaticamente em WSDL, é proposta a execução de um conjunto de etapas, utilizando as tecnologias discutidas no Capítulo 2, que transformarão um documento contendo informações sintáticas a respeito das funcionalidades de um *Web Service restful* em um documento semântico. Essas etapas, ilustradas na Figura 4-1, são:

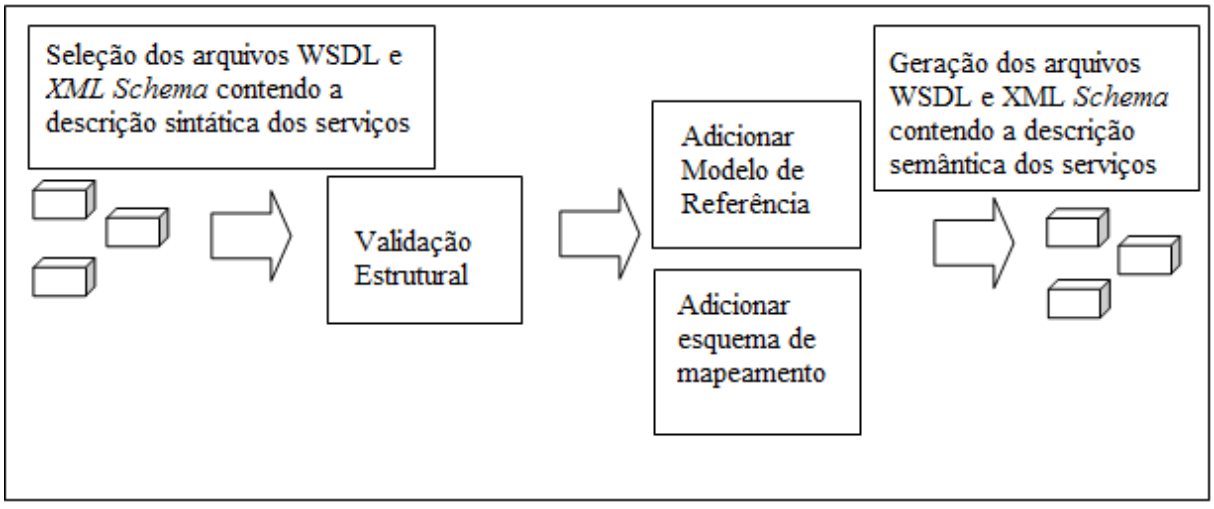
a) seleção dos arquivos WSDL ou *XML Schema Definition*, que poderão ser obtidos a partir de um servidor local ou de um servidor existente na web, conforme será discutido na Seção 4.3.2;
b) validação estrutural, esses arquivos terão a sua estrutura validada através da utilização do Serviço *Structural Validation*, discutido na Seção 4.3.1;

c) fornecer semântica aos arquivos WSDL *ou XML Schema Defintion* para inclusão dos atributos de extensão Modelo de Referência *(modelreference)* e o esquema de mapeamento *(liftingSchemaMapping* e *loweringSchemaMapping)* de acordo a especificação SAWSDL discutida no Capítulo 2. A especificação SAWSDL prevê a inclusão do atributo de referência *modelreference* no documento WSDL nos seguintes elementos (*interface*, *operation* e *fault*) e nas definições de tipo (e.g.: declarações de elementos, declarações de atributos) e a inclusão dos atributos de extensão *liftingSchemaMapping* e *loweringSchemaMapping* às declarações de elemento do *XML Schema Defintion* e definições de tipo simples e complexo. A anotação semântica de um tipo complexo pode ser realizada através de duas técnicas: (a) *Bottom Level*: anotação semântica realizada no nível do elemento ou atributo e (b) *Top Level*: anotação semântica realizada no nível da marcação do tipo complexo. Essas técnicas são discutidas na Seção 4.3.5;

d) geração dos arquivos WSDL ou *XML Schema Definition* contendo a descrição semântica. Espera-se que desenvolvedores de *software* em vez de realizar a anotação semântica manualmente utilizem a ferramenta SWS Editor, flexibilizando e aumentando a produtividade na execução dessa tarefa.

Além da possibilidade de realizar a anotação semântica nos documentos que contém a descrição sintática dos Serviços Web *restful* e nos documentos *XML Schema Definition*, também estão contemplados na solução a disponibilização de repositórios que mantém os Serviços Web *restful* semânticos. Os objetivos e funcionamento desses repositórios são detalhados na Seção 4.2.

Figura 4-1 Etapas para realizar anotação semântica no SWS Editor



Nas próximas seções são discutidas a arquitetura da ferramenta, as suas camadas, os serviços oferecidos pelo SWS Editor e as suas implementações com as tecnologias utilizadas.

4.2 ARQUITETURA DA FERRAMENTA SWS EDITOR

Na literatura especializada da Engenharia de *Software*, e.g. Pressman (2007) e Sommerville (2007) existe uma gama de conceitos sobre o processo de projeto de arquitetura, no qual envolvem o estabelecimento de um *framework* que identifica os principais componentes de um sistema e as suas comunicações. Sommerville (2007) apresenta três vantagens ao projetar e documentar uma arquitetura de *software*:

a) Comunicação de *stakeholders*[75]: O projeto de arquitetura de um sistema pode ser utilizado para gerar uma discussão entre os diferentes *stakeholders*;

b) Análise de Sistema: Com o objetivo de tornar a arquitetura do sistema explícita na fase inicial do desenvolvimento tornando mais fácil as tomadas de decisões arquiteturais que produzirão impacto no atendimento de alguns requisitos como desempenho e facilidade de manutenção;

c) Reutilização: A arquitetura de um sistema é uma descrição de como um sistema está organizado sendo que sistemas com requisitos similares podem possuir o mesmo projeto de arquitetura e que consequentemente apoia ao reuso em larga escala.

A arquitetura da ferramenta proposta busca oferecer as condições para que essas vantagens citadas por Sommerville possam ser estabelecidas através da utilização de Padrões de Projeto e Arquiteturais, e.g. MVC, e o uso de tecnologias, e.g. Serviços Web. A utilização desses padrões e tecnologias promovem o reuso de *software*. Sommerville (2007) cita os benefícios do reuso de *software*:

a) Confiança aumentada: o aplicativo que já foi testado e utilizado anteriormente possui uma confiabilidade maior do que aplicativos novos, pois os seus defeitos já foram encontrados e corrigidos;

b) Risco de processo reduzido: o custo de um aplicativo existente já é conhecido enquanto que os custos de um aplicativo a ser construído sempre trarão um problema de avaliação. Esse fator é importante no gerenciamento de projetos, pois a utilização de um *software* existente reduz a margem de custo do projeto;

c) Conformidade com padrões: alguns padrões, como por exemplo, os padrões de interface com o usuário, podem ser construídos como um conjunto de componentes

[75] Termo utilizado para identificar as partes interessados em diversas áreas como, por exemplo, na Arquitetura de Software.

reusáveis. O uso dessas interfaces padronizadas melhora a confiança, pois os usuários cometerão menos erros ao utilizar uma interface familiar;

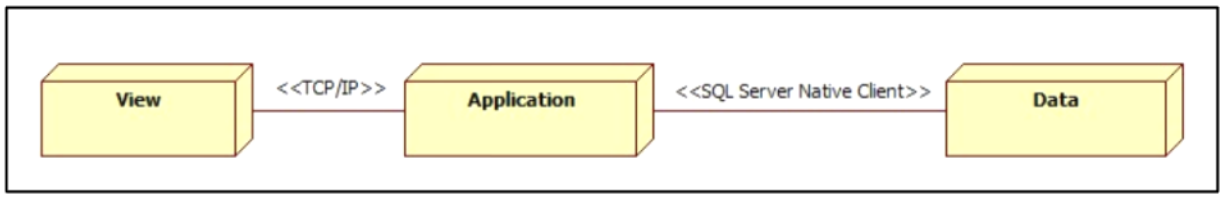
d) Desenvolvimento rápido: o reuso de software torna rápida a produção de *software*, pois o tempo para a realização do desenvolvimento e de testes deve ser reduzido.

A ferramenta proposta foi projetada a partir do desenvolvimento de serviços, pois de acordo com Choi, Nazareth e Jain (2010) a construção de *software* com a utilização de serviços fornece mais agilidade, eficiência e rapidez nas respostas ao ambiente de negócios das organizações.

Na arquitetura definida neste trabalho serão anotados semanticamente, a partir da utilização de serviços, os Serviços Web *restful* que possuem seus elementos descritos sintaticamente em WSDL 2.0 e a ligação dos conceitos ontológicos para cada elemento do documento WSDL 2.0 será realizado através da especificação SAWSDL. Os conceitos ontológicos são definidos por OWL (*Web Ontology Language*)[76].

A partir desses pressupostos, a definição da arquitetura da ferramenta foi baseada em um modelo arquitetural de camadas, conforme ilustrado na Figura 4-2, cujas interligações ocorrem por meio de tecnologias específicas para a comunicação entre as camadas, nesse caso TCP/IP[77] e *SQL Server Native Client*[78].

Figura 4-2 Arquitetura da ferramenta SWS EDITOR



Esta arquitetura está ilustrada com maior detalhe na Figura 4-3 que possui a camada de Visão (*View)* correspondente a *View* da Figura 4-2. A camada identificada como *Application*, ilustrada na Figura 4-2, corresponde a camada *Application* da Figura 4-3, que contém os serviços responsáveis em fornecer o mecanismo para a anotação semântica dos Serviços Web. A camada de repositório de dados (*Data*) da Figura 4-2 se refere à camada de dados (*Data*) da

[76] http://www.w3.org/2001/sw/wiki/OWL

[77] http://global.britannica.com/EBchecked/topic/602945/TCPIP

[78] http://technet.microsoft.com/pt-br/library/ms190944(v=sql.105).aspx

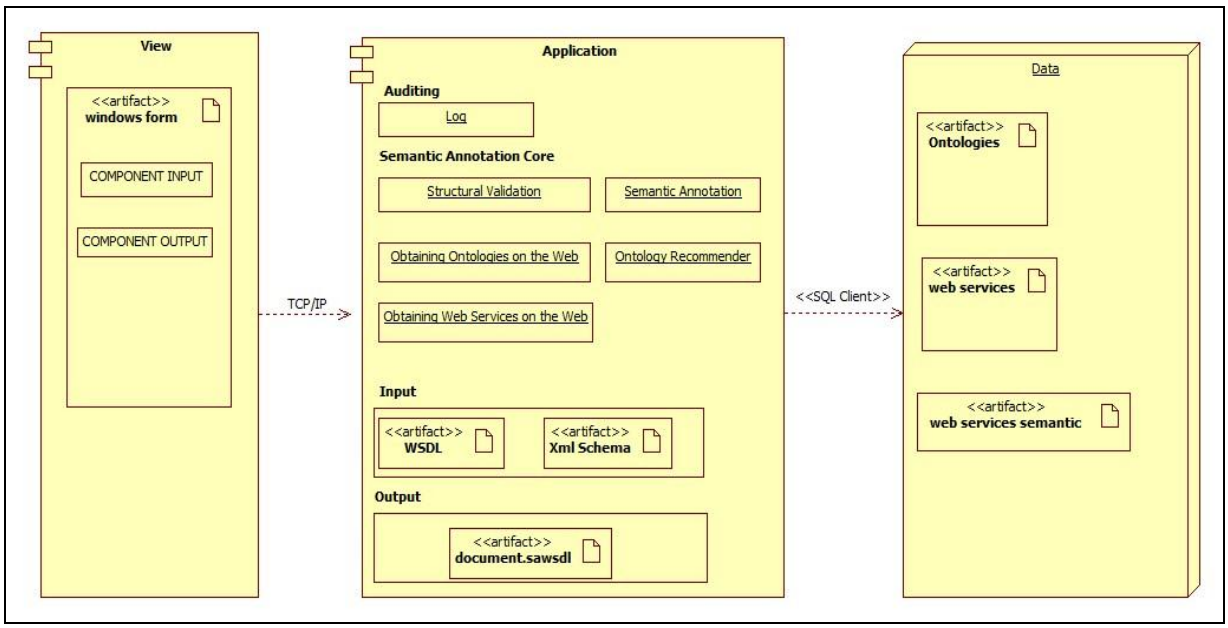
Figura 4-3, que corresponde ao repositório de dados que armazenarão as ontologias OWL e Serviços Web semânticos

A camada de Visão (*View*), ilustrada na Figura 4-3, desempenha o papel da interface entre o consumidor e a ferramenta no qual são disponibilizadas as funcionalidades, estabelecendo o relacionamento entre o cliente e os serviços oferecidos pela ferramenta. O componente identificado na Figura 4-3 como COMPONENT INPUT obtém os dados de entrada que são necessários para realizar a anotação semântica dos *Web Service restful*, através do usuário da ferramenta que seleciona os documentos *XML Schema* ou WSDL em disco ou na web e o COMPONENT OUTPUT fornece os dados de saída para o usuário contendo esses documentos anotados semanticamente em disco contendo a extensão SAWSDL.

A comunicação entre as camadas, identificada através das setas de associação, ocorre com a utilização do protocolo TCP/IP entre a camada de visão (*View*) e a camada de aplicação (*Application*) e com a utilização da API *SQL Server Native Client* entre a camada de aplicação (*Application*) e a camada de dados (*Data*), conforme ilustrado na Figura 4-3.

Na camada *Application* estão os serviços oferecidos pela ferramenta e publicados através de um servidor de aplicação. Esses serviços atuando em conjunto são responsáveis em prover a anotação semântica dos Serviços Web *restful*.

Figura 4-3 Detalhamento da Arquitetura da Ferramenta SWS Editor

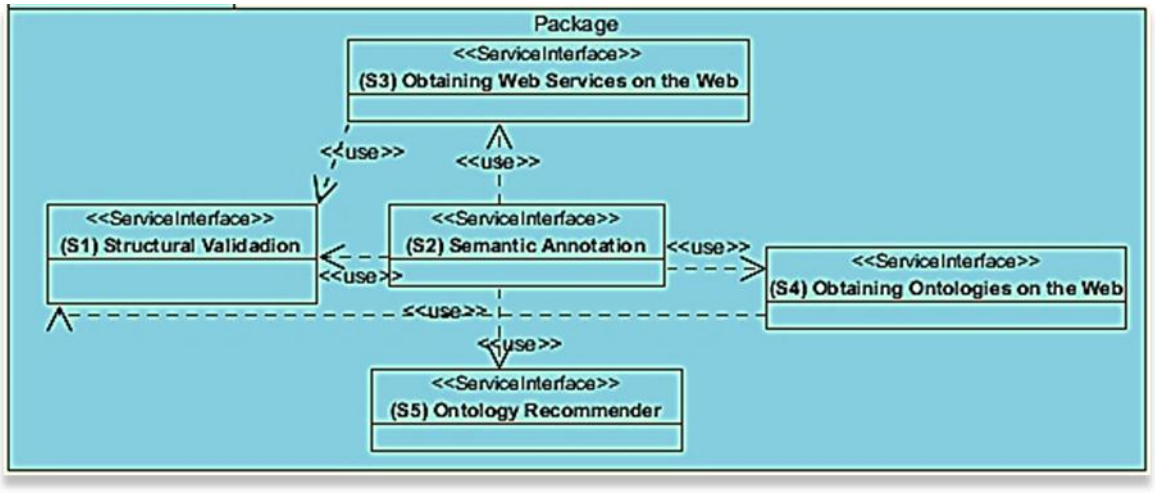


Na camada *Application* é possível observar na Figura 4-3 o componente que representa a auditoria (*Auditing*) na parte superior e dois componentes (*input* e *output*) que representam os arquivos utilizados como entrada de dados (WSDL e *XML Schema*) e o

formato de saída (document.sawsdl) dos arquivos gerados pelos serviços oferecidos pela ferramenta. Na Figura 4-3 consta o núcleo de anotação semântica representada pelos seguintes serviços: (S1) *Structural Validation*: possui o objetivo de realizar a análise dos documentos WSDL e *XML Schema Definition* no que se refere a sua estrutura e formação; (S2) *Semantic Annotation*: possui o objetivo de fornecer anotação semântica aos Serviços Web *restful*; (S3) *Obtaining Web Services on the Web*: este serviço tem como objetivo obter os Serviços Web *restful* que estejam localizados na Internet; (S4) *Obtaining Ontologies on the Web*: este serviço tem como objetivo obter ontologias que estejam disponíveis na internet; (S5) *Ontology Recommender*: a responsabilidade deste serviço é fornecer um recomendador de ontologias para realizar anotação de semântica de Serviços Web similares aos já anotados pela ferramenta.

De acordo com Erl (2009) a disponibilização de serviços favorece ao baixo acoplamento, e.g., a fraca dependência na comunicação entre os serviços, e a capacidade de reuso, que possibilita condições para que diferentes ambientes possam compartilhar os mesmos recursos. A Figura 4-4 ilustra o relacionamento existente entre os serviços oferecidos pela ferramenta proposta, no qual os serviços S2, S3 e S4 utilizam o serviço S1, enquanto S3, S4 e S5 fazem composição com S2. Esses serviços serão detalhados nas Seções 4.3 e 4.4.

Figura 4-4 Relacionamento entre os serviços



Na definição da ferramenta SWS Editor, esses serviços implementam os seguintes requisitos:

(R1) Anotação semântica de Serviços Web *restful* descritos sintaticamente com WSDL 2.0, e (R2) Anotação semântica de Serviços Web no padrão REST implementados no serviço (S2) *Semantic Annotation* e (S1) *Structural Validation*;

(R3) Utilização de Ontologias preexistentes na Internet, implementado no serviço (S4) *Obtaining Ontologies on the Web*;

(R4) Utilização de tecnologias que são recomendações do W3C, todos os serviços utilizados no SWS Editor para realizar a anotação semântica utilizam tecnologias que são recomendadas pelo W3C;

(R5) Utilizar conceitos ontológicos definidos em OWL, implementado pelo serviço (S4) *Obtaining Ontologies on the Web* e (S5) *Ontology Recommender*;

(R6) Manter um repositório de serviços web semânticos e (R7) Manter um repositório de Ontologias para processar as anotações de um documento WSDL e/ou XML *Schema Definition*, implementados a partir do uso dos serviços (S2) *Semantic Annotation*, (S3) *Obtaining Web Services on the Web*, (S4) *Obtaining Ontologies on the Web*; (R9) Realizar a anotação de Serviços Web construídos de acordo com o padrão SOAP, implementado pelos serviços (S1) *Structural Validation*, (S2) *Semantic Annotation* e *Obtaining Web Services on the Web* sendo que os demais serviços da ferramenta SWS Editor possuem suporte para a anotação semântica de Serviços Web SOAP;

R8) Permitir a extensibilidade da ferramenta através da disponibilização de serviços, é a solução elaborada por este trabalho, a partir da utilização de serviços, para o fornecimento de semântica aos Serviços Web.

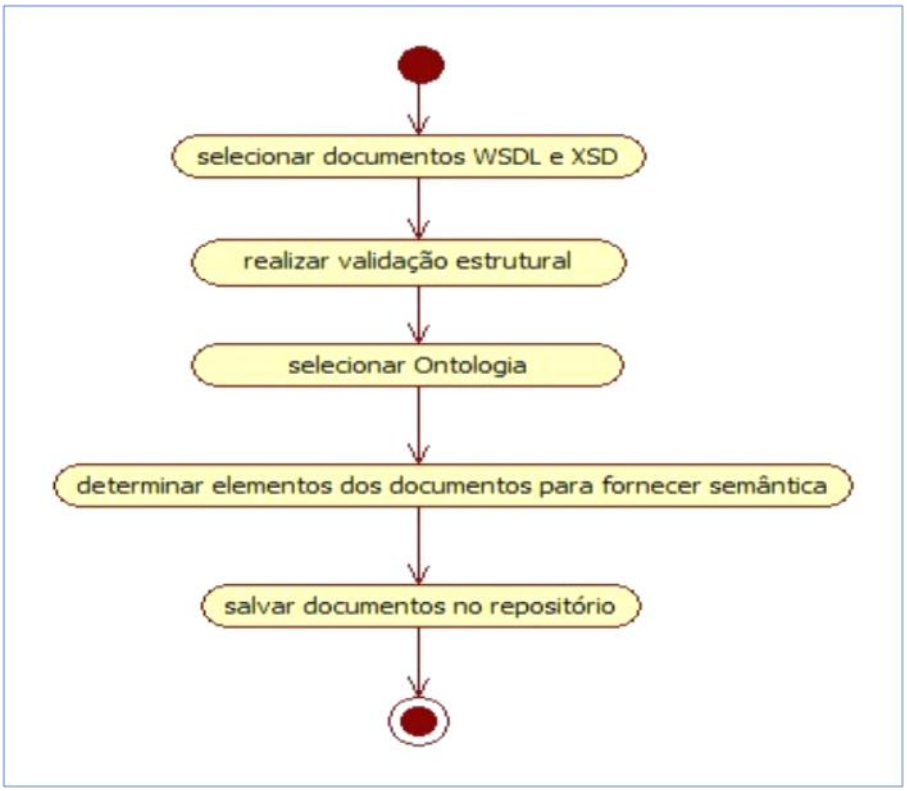
Cada serviço oferecido pela ferramenta possui um contrato, representado pelo arquivo WSDL, no qual está descrito como acessá-lo e quais métodos e operações estão disponíveis. A capacidade de reuso discutida no Capítulo 2 está presente nos serviços, pois poderão ser reutilizados por quaisquer outros ambientes que necessitem realizar a anotação semântica de Serviços Web, não ficando limitado apenas ao ambiente *desktop*.

Esses serviços podem fazer uso de um repositório de dados na execução de suas tarefas. Esse recurso está representado na terceira camada do modelo proposto para a ferramenta, no qual são inseridos dados como: (i) arquivos WSDL, SAWSDL e Ontologias (ii) definições para autenticação e autorização de acesso, (iii) tratamento de mensagens de erro e (iv) informações de auditoria (objetivo do serviço, restrições de uso, versão, desenvolvedor, informações de contato, entre outras).

A Figura 4-5 mostra o diagrama de atividades do processo de anotação semântica dos documentos XML *Schema Defintion* e WSDL realizada pela ferramenta SWS Editor. O processo é iniciado com a seleção dos documentos WSDL 2.0 ou *XML Schema Definition*

pelo usuário da ferramenta, sendo que a seleção desses documentos poderá ser realizada através de duas formas: (i) obtenção dos documentos armazenados em um servidor local e (ii) obtenção do WSDL na web. Na sequência o usuário seleciona a ontologia, definida em linguagem OWL, para determinar o vocabulário semântico dos elementos nos documentos selecionados anteriormente. A seleção da ontologia pelo usuário da ferramenta pode ser realizada através de três formas: (i) obtenção da ontologia armazenados em um servidor local; (ii) obtenção da ontologia na web; e (iii) recomendador de ontologia. O vocabulário semântico é dado pelas inclusões dos atributos de extensão: (i) *modelreference*, (ii) *liftingSchemaMapping* e *loweringSchemaMapping*. Esses atributos de extensão, da especificação SAWSDL, foram discutidos no Capítulo 2 e a utilização desses atributos pela ferramenta SWS Editor será discutido nas Seções 4.3.5.1, e 4.3.5.2. O processo é finalizado com o documento WSDL contendo o vocabulário semântico.

Figura 4-5 Atividades do processo de anotação semântica do SWS Editor



Os serviços disponíveis na ferramenta, seus objetivos, tecnologias e detalhes da sua construção são descritos nas Seções 4.3 e 4.4.

4.3 SERVIÇOS DO SWS EDITOR

Um modelo de objetos é utilizado para projetar os requisitos dos sistemas utilizando uma linguagem de programação orientada a objetos (SOMMERVILLE, 2007). De acordo com Larman (2007) um modelo de domínio não é apenas uma coleção de diagramas, mas uma representação visual de objetos do mundo real através da implementação de classes. Por essa razão, cada serviço disponibilizado no SWS Editor foi concebido sob uma estrutura de modelo de objetos contendo classes inter-relacionadas que favorece o reuso e a redução do esforço no desenvolvimento de sistemas.

A estratégia para a elaboração dos requisitos dos serviços oferecidos pela ferramenta SWS Editor foi determinada no Capítulo 3 e os requisitos para a construção dos seus serviços seguiram alguns princípios apresentados por Fugita e Hirama (2012) e.g. (i) contrato padronizado: os documentos formais para descrever o contrato do serviço (e.g..: interfaces WSDL) foram criados em conformidade com as boas práticas recomendadas pelo W3C (e.g.: padrões técnicos de interoperabilidade, padrões de formato); (ii) reusabilidade: o acesso ao serviço é realizado somente através de seu contrato e poderá ser facilmente acessado por diferentes ambientes (e.g.: aplicações web, *mobile*); (iii) visibilidade: o contrato do serviço deve ser disponibilizado para potenciais usuários ou outros serviços e para isso o contrato estará disponível em um repositório. No estágio atual deste trabalho foi utilizado como repositório de serviços o servidor web IIS (*Internet Information Service*)[79] que foi utilizado no desenvolvimento e publicação desses serviços; (iv) Interoperável: os artefatos técnicos (e.g.: WSDL, XML *Schema Definition*) foram elaborados utilizando padrões abertos (e.g.: XML).

Nas próximas seções serão discutidos os serviços responsáveis em fornecer semântica aos elementos dos documentos WSDL e *XML Schema Definition.*

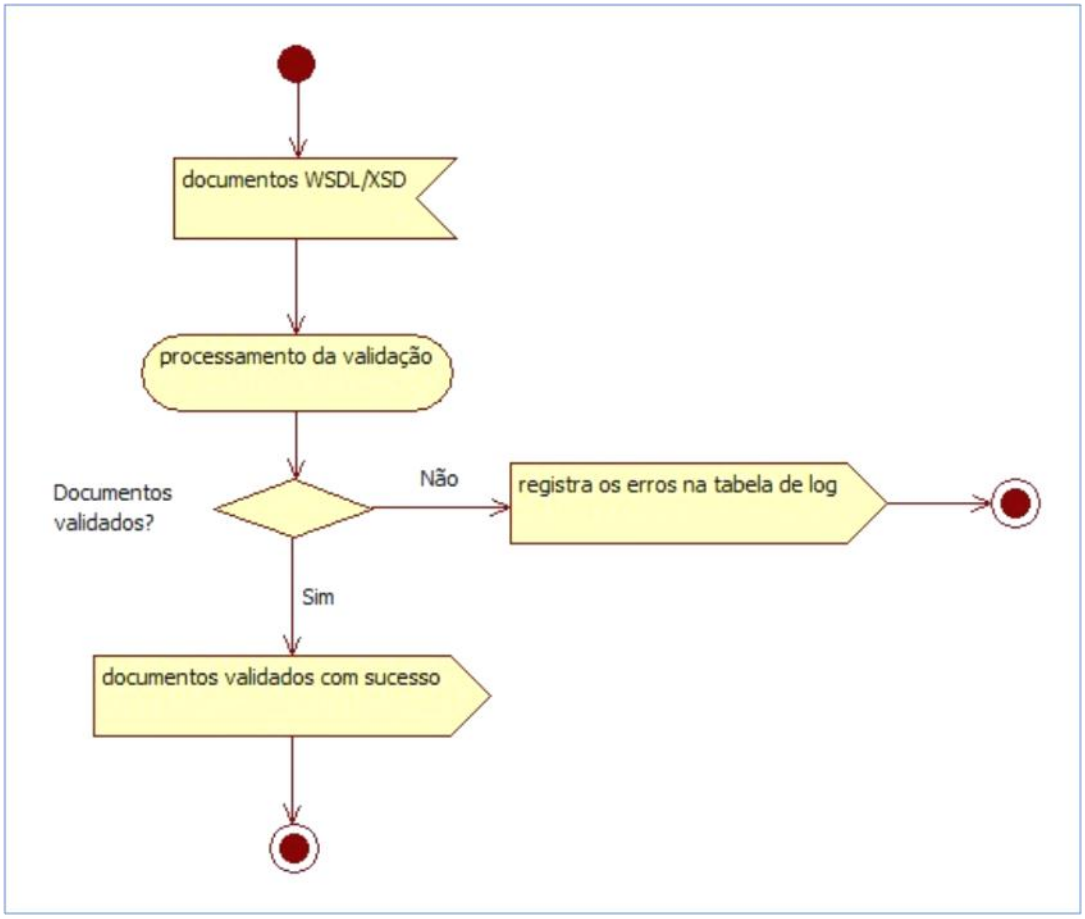
4.3.1 *Structural Validation*

O serviço *Structural Validation* tem por objetivo analisar a estrutura dos documentos WSDL e *XML Schema Definition* (instância, formação), a fim de verificar se essas estruturas foram definidas de acordo com as especificações da linguagem XML da W3C e da obrigatoriedade dos elementos existentes nesses documentos com seus respectivos elementos e atributos. A Figura 4-6 mostra o diagrama de atividades do serviço de validação estrutural, cujo processo é iniciado com o recebimento do documento WSDL ou do *WSDL Schema Definition*, selecionados pelo usuário da ferramenta.

---

[79] https://technet.microsoft.com/pt-br/library/hh831725.aspx

Figura 4-6 Diagrama de Atividades do serviço *Structural Validation*



Após o recebimento dos documentos, o procedimento de validação estrutural dos documentos WSDL ou *XML Schema Defintion* é executado no SWS Editor. Se forem detectados quaisquer problemas após o procedimento de validação, serão retornadas as mensagens de erro para o serviço da ferramenta, o fluxo do processo registra a mensagem de erro no repositório de dados e outra mensagem é apresentada ao usuário da ferramenta sendo que nesse momento o processo é encerrado. Exemplos das mensagens de erros identificados pelo sistema são: (i) “Documento WSDL mal formatado - Necessário realizar a correção no documento” e (ii) "<stackTrace> System.InvalidOperationException: There is an error in XML document (1, 402) at utilitario. _ValidatorXML.validarXml(String Xml) </stacktrace>". Nesse caso, a mensagem (i) é apresentada em tela para o usuário e a mensagem (ii) é armazenada no repositório para posterior consultas desses erros pelos desenvolvedores de *software*.

Por outro lado, se os procedimentos no fluxo do processo de validação ocorrerem sem apresentar quaisquer problemas, a execução é continuada com o envio da mensagem ao usuário da ferramenta, informando sobre a corretude dos documentos tendo em vista a validação ter sido realizada com sucesso.

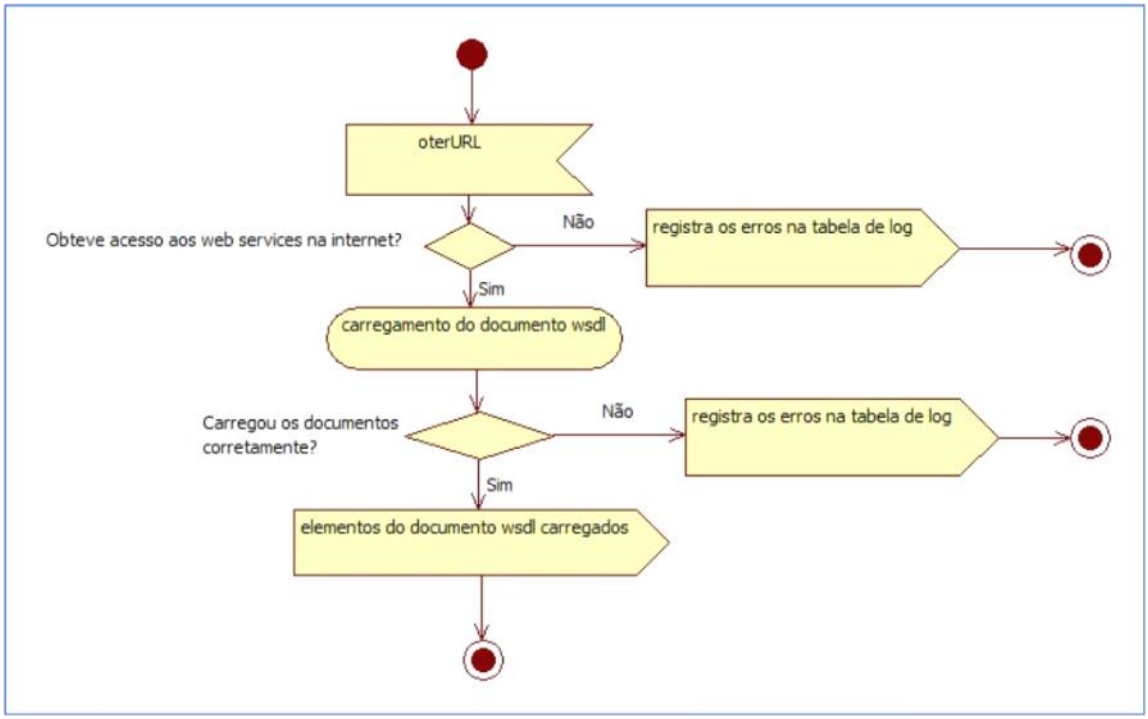
4.3.2 *Obtaining Web Services on the web*

Este serviço tem como objetivo obter os Serviços Web que estejam disponíveis na Internet. O usuário da ferramenta SWS Editor é responsável em realizar o fornecimento da URL do *Web Service* que necessita da anotação semântica disponível em um repositório na web. Essa obtenção é concluída com o *downloading* do documento WSDL e feita a carga dos seus elementos por meio da ferramenta usando uma estrutura de árvore.

A Figura 4-7 mostra o processo sendo iniciado a partir do fornecimento da URL. Em seguida, é feita a carga do documento apresentando todos os elementos do WSDL ao usuário da ferramenta.

Se ocorrerem erros durante o *downloading* do documento WSDL ou na carga dos elementos do documento WSDL, o fluxo do processo armazena a mensagem de erro no repositório de dados e uma mensagem de erro é apresentada ao usuário da ferramenta sendo que nesse momento o processo é encerrado. Por outro lado, não havendo a existência de erros, o processo continua com a carga dos elementos do documento WSDL. Os detalhes desse serviço são demonstrados na Seção 4.4.2, correspondente a implementação deste serviço.

Figura 4-7 Diagrama de Atividade para o serviço *Obtaining Web Services on the web*



Após a execução desse procedimento, o *Web Service* pode ser anotado semanticamente utilizando o serviço anotador semântico.

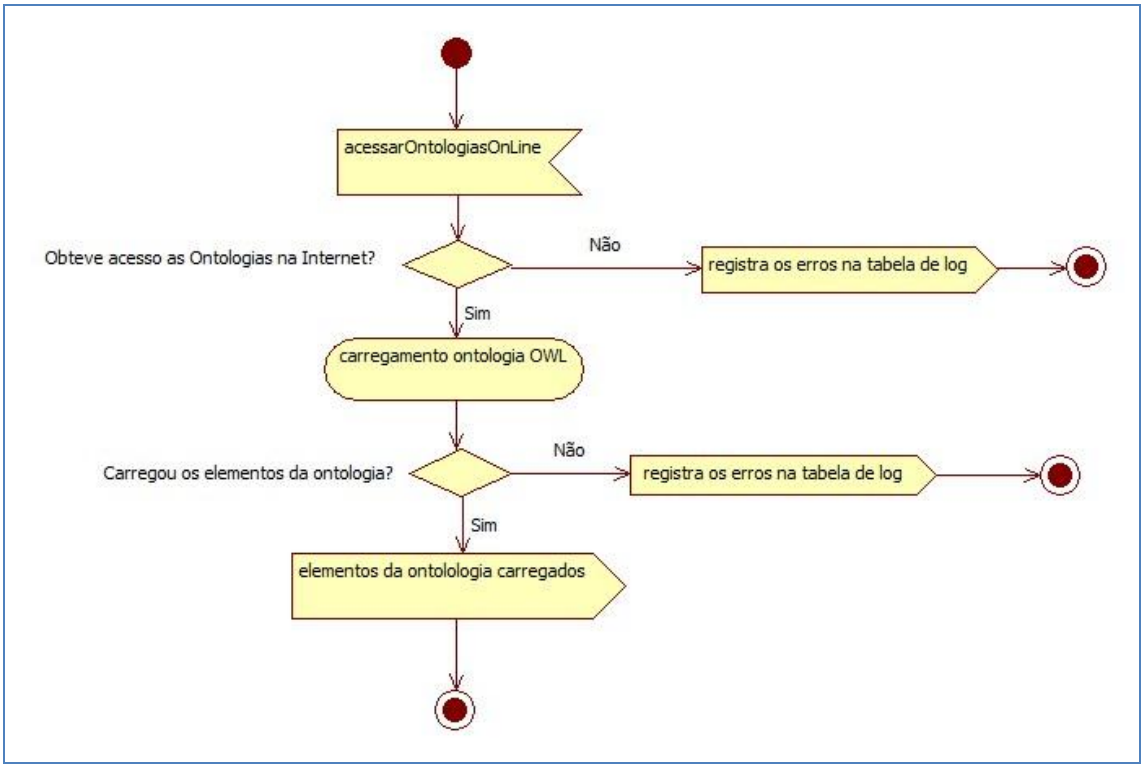
4.3.3 *Obtaining Ontologies on the web*

Este serviço tem como objetivo obter ontologias que estejam disponíveis na Internet. O usuário da ferramenta SWS Editor é responsável em realizar o fornecimento da URL onde está localizada a Ontologia OWL. Essa obtenção é concluída com a carga dos elementos do documento OWL por meio da ferramenta.

A Figura 4-8 mostra o processo sendo iniciado a partir do fornecimento da URL da ontologia existente na Internet. Em seguida, o documento apresenta todos os elementos da Ontologia descrita em linguagem OWL.

Se ocorrerem erros durante o acesso da ontologia OWL na Internet ou no procedimento de carregamento do documento OWL na ferramenta, o fluxo do processo armazena a mensagem de erro no repositório de dados e uma mensagem de erro é apresentada ao usuário da ferramenta sendo que nesse momento o processo é encerrado. Por outro lado, não havendo a existência de erros, o processo segue com a carga dos elementos da Ontologia por meio ferramenta SWS Editor em uma estrutura de árvore.

Figura 4-8 Diagrama de Atividades para o serviço *Obtaining Ontologies on the web*



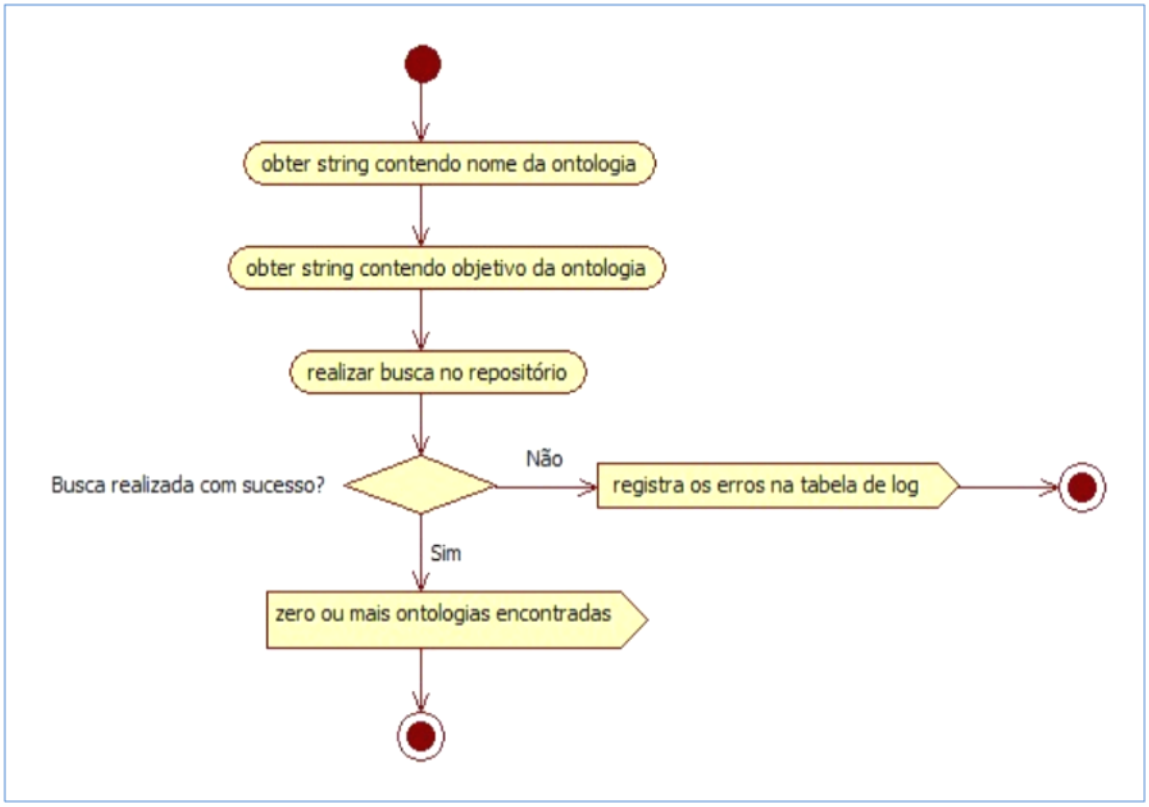
Após a execução desse procedimento, a ontologia OWL estará disponível para fornecer a anotação semântica nos Serviços Web *restful*. A anotação semântica dos Serviços Web é realizada pelo serviço *Semantic Annotation* discutido na Seção 4.3.5.

4.3.4 *Ontology Recommender*

Este serviço tem como objetivo implementar um recomendador de ontologias OWL para realizar a anotação de semântica de Serviços Web similares aos já anotados pela ferramenta. Nesse serviço o usuário fornece uma *string* que representa o nome da ontologia e/ou outra que descreve o seu objetivo através do qual obtém a ontologia desejada.

A Figura 4-9 mostra o processo sendo iniciado a partir do fornecimento pelo usuário da *string* que representa o nome da ontologia e logo após deverá ser fornecido a *string* que descreve o objetivo da ontologia. A partir desses parâmetros, a SWS Editor realiza a busca no repositório de ontologia e caso localize a ontologia retorna à informação para o usuário que poderá iniciar o processo de anotação semântica dos elementos dos documentos WSDL e XML *Schema Definition* a partir da utilização do serviço *Semantic Annotation* discutido na Seção 4.3.5.

Figura 4-9 Diagrama de Atividades para o serviço *Ontology Recommender*



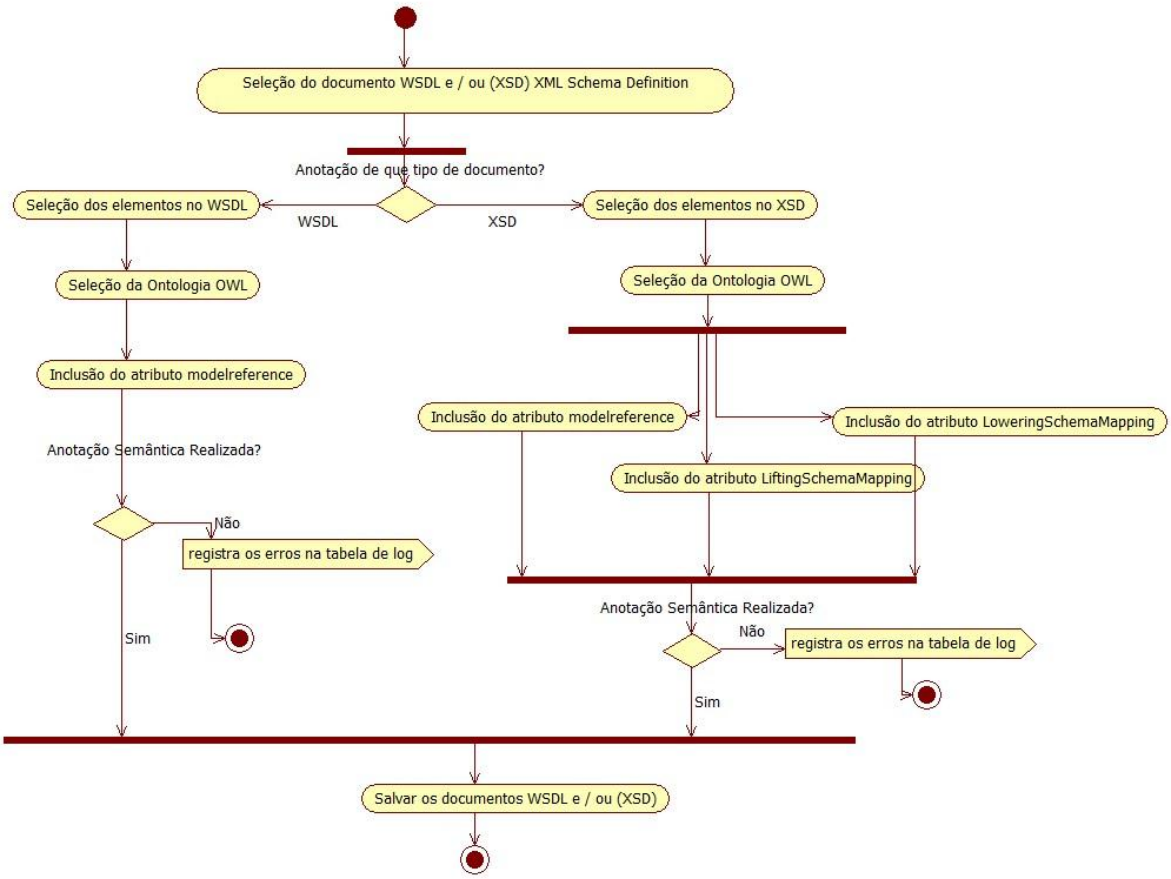
4.3.5 *Semantic Annotation*

O Serviço *Semantic Annotation* tem como objetivo fornecer a anotação semântica dos Serviços Web *restful* com o auxílio dos serviços discutidos nas Seções 4.3.2, 4.3.3 e 4.3.4. Para realizar a anotação semântica desses serviços que estão descritos em documentos WSDL 2.0 foi utilizado a especificação SAWSDL. Conforme discutido no Capítulo 2, a especificação SAWSDL utiliza o documento WSDL para fornecer a semântica aos Serviços Web. A utilização da SAWSDL permite que os desenvolvedores de *software* anotem semanticamente os Serviços Web sem precisar aprender uma linguagem diferente da WSDL. O vocabulário semântico, utilizado pela especificação SAWSDL, para fornecer significado aos documentos WSDL 2.0 é determinado pelas ontologias que são definidas em linguagem OWL, discutida no Capítulo 2 desta dissertação.

A Figura 4-10 mostra o processo sendo iniciado com o recebimento dos documentos XML (WSDL 2.0, XML *Schema Definition*) após sua validação sintática, ilustrada na Figura 4-6. A seleção desses documentos na ferramenta poderá ocorrer de duas formas: (i) o usuário seleciona o documento WSDL e/ou XML *Schema Definition* armazenado em repositório local ou de rede; ou (ii) a seleção do documento WSDL poderá ocorrer a partir do fornecimento da URL do *Web Service* existente na Internet utilizando o serviço *Obtaining Web Services on the*

web discutido na Seção 4.3.2. Em seguida, esses documentos são anotados semanticamente a partir da seleção da ontologia apropriada para determinar o vocabulário semântico do *Web Service*. A seleção da ontologia poderá ocorrer a partir de três opções: (i) o usuário da ferramenta seleciona a ontologia OWL armazenada em repositório local ou de rede; (ii) a seleção da ontologia OWL poderá ocorrer a partir do fornecimento da URL da ontologia existente na Internet utilizando o serviço *Obtaining Ontologies on the web* discutido na Seção 4.3.3 e (iii) a seleção da ontologia OWL poderá ocorrer a partir da utilização do serviço *Ontology Recommender* discutido na Seção 4.3.4. Após a seleção da ontologia, o usuário da SWS Editor poderá realizar a anotação dos elementos dos documentos WSDL 2.0 e XML *Schema Definition* e nas Seções 4.3.5.1 e 4.3.5.2 são discutidos a anotação semântica realizada pelo SWS Editor nesses documentos.

Figura 4-10 Diagrama de Atividades para o serviço *Semantic Annotation*



4.3.5.1 Anotando documentos WSDL no SWS Editor

A SAWSDL especifica os elementos do WSDL que podem receber a anotação semântica através do atributo de extensão *modelreference*: (i) *interface*; (ii) *operation* e (iii) *fault*. No caso da anotação semântica realizada no documento WSDL 2.0, a SWS Editor

adiciona o atributo (*sawsdl:modelReference*) nos elementos citados nos itens (i), (ii) e (iii). A Listagem 4-1 exemplifica o trecho de um documento WSDL que utiliza o *modelreference* para anotar semanticamente e categorizar o elemento *interface*. O *modelReference* neste exemplo aponta para os filmes classificados como romântico no modelo semântico filme.owl.

Listagem 4-1 Trecho do elemento *interface* anotado semanticamente no SWS Editor

```
1.<wsdl:interface name="publicacao"
2.sawsdl:modelReference="http://example.edu.unifacs/ontologies/filme.owl#romantico">
3. …
4.</wsdl:interface>
```

A Listagem 4-2 exemplifica o trecho de um documento WSDL que utiliza o *modelreference* para anotar semanticamente o elemento *operation* e nesse caso fornece uma descrição da operação do serviço (e.g.: parâmetro de entrada, tipo de resposta). O *modelReference* neste exemplo utiliza a classe ComprarLivro da ontologia livro.owl para descrever o elemento *operation* denominado realizarCompra.

Listagem 4-2 Trecho do elemento operation anotado semanticamente no SWS Editor

```
1.<wsdl: operation name="realizarCompra" pattern="http://www.w3.org/ns/wsdl/in-out"
2.<sawsdl: modelReference="http://example.edu.unifacs/ontologies/livro.owl#ComprarLivro">
4.  <wsdl:input element="LivroRequest"/>
5.  <wsdl:output element="LivroResponse"/>
6.</wsdl: operation>
```

A Listagem 4-3 exemplifica o trecho de um documento WSDL que utiliza o *modelreference* para anotar semanticamente o elemento *fault* e nesse caso fornece uma descrição de falha (e.g.: exceções ocorridas na solicitação do serviço). O *modelReference* neste exemplo identifica a classe LivroInexistente da ontologia livro.owl para descrever a falha LivroInexistenteErro.

Listagem 4-3 Trecho do elemento fault anotado semanticamente no SWS Editor

```
1.…
2.<wsdl:interface name="ComprarLivro">
3.  <wsdl:fault name="LivroInexistenteErro" element="AvailabilityInformation"
4.sawsdl:modelReference="http://
5.example.edu.unifacs/ontologies/livro.owl/comprarlivro#LivroInexistente"/>
6.…
7.</wsdl:interface>
```

4.3.5.2 Anotando documentos *XML Schema Definition* no SWS Editor

A SAWSDL especifica dois mecanismos (*modelreference* e *schemamapping*) para realizar a anotação semântica dos elementos (e.g.: tipos simples, tipos complexos, elementos, atributos) dos documentos XML *Schema Definition.* Da mesma forma como ocorre com o procedimento de anotação semântica nos elementos do documento WSDL, a SWS Editor adiciona um novo atributo (*sawsdl*) aos elementos dos documentos XML *Schema Definition* para fornecer o vocabulário semântico a esses documentos. A Listagem 4-4 exemplifica o trecho de um documento XML *Schema Definition* que utiliza o *modelreference* para anotar semanticamente o elemento (*simpleType*). Neste exemplo, quaisquer elementos ou atributos cujo tipo de dado é Data serão descritos pelo conceito UTCData no modelo semântico Util.owl.

Listagem 4-4 Trecho do elemento simpletype anotado semanticamente no SWS Editor

```
1. <xs:simpleType name="Data"
2.sawsdl:modelReference="http://example.edu.unifacs/ontologies/Util.owl#UTCData">
3.  …
4.</xs:simpleType>
```

A anotação semântica de um *complexType* pode ser realizada através de duas técnicas:

a) *Bottom Level*: A anotação semântica é realizada no nível do elemento ou atributo. A Listagem 4-5 exemplifica o uso dessa técnica. Neste exemplo, cada elemento de nível

inferior possui uma anotação semântica uma vez que a ontologia Util.owl contém os conceitos para os elementos “valor” e “código”.

b) *Top Level*: A anotação semântica é realizada no nível da marcação complexType. A Listagem 4-6 exemplifica o uso dessa técnica. Nesse caso, o complexType foi anotado semanticamente através do qual os elementos “valor” e “codigo” possuem os seus conceitos determinados pelo mesmo conceito ontológico (Valores).

Listagem 4-5 Trecho do elemento *complexType* anotado semanticamente (*Bottom Level*) no SWS Editor

```
1.…
2.<xs:complexType>
3.  <xs:sequence minOccurs="1" maxOccurs="9">
4.    <xs:element name="valor" type="xs:decimal"
5.sawsdl:modelReference="http://example.edu.unifacs/ontologies/Util.owl#Valor"/>
6.    <xs:element name="codigo" type="xs:string"
7.sawsdl:modelReference=" http://example.edu.unifacs/ontologies/Util.owl#Codigo"/>
8.  </xs:sequence>
9.</xs:complexType>
10.…
```

Listagem 4-6 Trecho do elemento *complexType* anotado semanticamente (*topLevel*) no SWS Editor

```
1…
2.<xs:complexType sawsdl:modelReference=" http://
3.example.edu.unifacs/ontologies//Util.owl#Valores">
4.  <xs:sequence minOccurs="1" maxOccurs="unbounded">
5.    <xs:element name="valor" type="xs:decimal"
6.    <xs:element name="codigo" type="xs:string"
7.  </xs:sequence>
8.</xs:complexType>
9…
```

A anotação semântica de elementos e atributos do XML *Schema Definiti*on ocorrem de maneira similar ao realizado nos exemplos demonstrados nas Listagens 4.4, 4.5 e 4.6.

SAWSDL realiza o mapeamento de esquema a partir da utilização de dois atributos de extensão: (i) o *liftingSchemaMapping* e o (ii) *loweringSchemaMapping*. A Listagem 4-7 exemplifica o trecho de um documento XML *Schema Defintion* que utiliza o atributo *liftingSchemaMapping* para especificar o mapeamento de esquema através do uso da linguagem XSLT. Nesse caso, o que foi programado no arquivo

http://www.w3.org/2002/ws/sawsdl/spec/mapping/Response2Ont.xslt será aplicado aos elementos EnderecoResponse (nomeRua, nomeCidade, nomeEstado).

Listagem 4-7 Mapeamento de esquema através do uso da linguagem XSLT

```
1.…
2.<xs:element name="EnderecoResponse" >
3.<xs:complexType
4.sawsdl:liftingSchemaMapping="http://www.w3.org/2002/ws/sawsdl/spec/mapping/Respons
5.e2Ont.xslt">
6.  <xs:sequence>
7.   <xs:element name=”nomeRua” type=”xs:string” />
8.    <xs:element name="nomeCidade" type=”xs:string” />
10.    <xs:element name ="nomeEstado" type=”xs:string”  />
11.  </xs:sequence>
12.</xs: complexType >
13.</xs:element>
14…
```

Se ocorrerem erros durante a anotação semântica dos elementos dos documentos WSDL e/ou XML *Schema Definition*, o fluxo do processo registra a mensagem de erro no repositório de dados e uma mensagem de erro é apresentada ao usuário da ferramenta sendo que nesse momento o processo é encerrado. Por outro lado, não havendo a existência de erros, o processo segue com a geração do documento XML de saída contendo as informações semânticas. Os detalhes de implementação desse documento semântico gerado são discutidos na Seção 4.4.5.

Na Seção 4.4 são discutidas as questões relacionadas à implementação da SWS Editor e os serviços disponibilizados, considerando as características de reuso, conformidade com padrões discutidos neste capítulo e aspectos tecnológicos como a plataforma de desenvolvimento utilizada na solução.

4.4 IMPLEMENTAÇÃO DOS SERVIÇOS OFERECIDOS PELA FERRAMENTA SWS EDITOR

A implementação dos serviços da ferramenta SWS Editor foi baseado na especificação *WS-I Basic Profile*[80] que consiste de um conjunto de princípios para o desenvolvimento de Serviços Web com o objetivo de promover a interoperabilidade, e.g. ser compatível com serviços implantados, foco na interoperabilidade, aplicações semânticas.

[80] http://ws-i.org/profiles/basicprofile-2.0-2010-11-09.html

O ambiente para o desenvolvimento foi composto da seguinte forma: (i) os diagramas de atividade, diagramas de classe e sequência foram elaborados na versão 5.0.2.1570 da ferramenta *StARUML*[81], uma ferramenta de modelagem de diagramas UML; (ii) para ilustrar o relacionamento entre os serviços foi utilizada a ferramenta *Visual Paradigm Enterprise* Edition na versão 12 que implementa a especificação *SoaML* (*Service-oriented architecture Modeling Language*)[82], um projeto *open-source* derivado da UML e mantido pela OMG (*Object Management Group*)[83] para a modelagem de serviços; (iii) para a codificação dos serviços foi utilizada a IDE *Microsoft Visual Studio*[84] na versão 2013 utilizando a linguagem vb.net e *framework* .Net 4.5; (iii) para a publicação dos Serviços Web, foi usado o servidor web IIS, (iv) para armazenar as ontologias OWL, documentos WSDL e *XML Schema Definition* utilizados no desenvolvimento deste trabalho foi utilizado o sistema gerenciador de banco de dados SQL Server 2014 Express[85] e (v) para definição do repositório semântico foi utilizado o *framework Jena* na versão 2.13.0, a linguagem de programação Java na versão 8 e a IDE Eclipse[86] na versão *Kepler Service Release 2.*

A implementação das classes faz uso de padrões de projeto como o *BLL* (*Business Logic Layer)* [87], *DAL* (*Data Access Layer)* [88] e o *Facade* [89] e padrões arquiteturais como o MVC com o objetivo de tornar a solução reutilizável. As classes implementadas como *Business Logic Layer* serão responsáveis em estabelecer as regras de negócio da ferramenta (i.e.: manipulação de XML, documentos OWL e WSDL) além de fornecer uma ponte entre a camada DAL e o banco de dados. Já a camada DAL tem for finalidade fornecer acesso aos dados armazenados em um banco de dados, ou seja, os comandos DML (*Data Manipulation Language*)[90], tais como *insert, update, delete* e *select*, da linguagem SQL (*Structured Query Language*)[91] deverão ser escritos em classes DAL. As classes implementadas como Facade

[81] http://sourceforge.net/projects/staruml/files/staruml/5.0/

[82] http://www.omg.org/spec/SoaML/

[83] http://www.omg.org/

[84] https://msdn.microsoft.com/pt-br/library/dd831853.aspx

[85] https://www.microsoft.com/pt-br/download/details.aspx?id=42299

[86] https://eclipse.org/kepler/

[87] https://msdn.microsoft.com/en-us/library/aa581779.aspx

[88] https://msdn.microsoft.com/en-us/library/aa581776.aspx

[89] https://msdn.microsoft.com/en-us/library/orm-9780596527730-01-04.aspx

[90] http://databases.about.com/od/sql/a/sqlfundamentals_3.htm

[91] http://databases.about.com/od/sql/u/learning_sql.htm

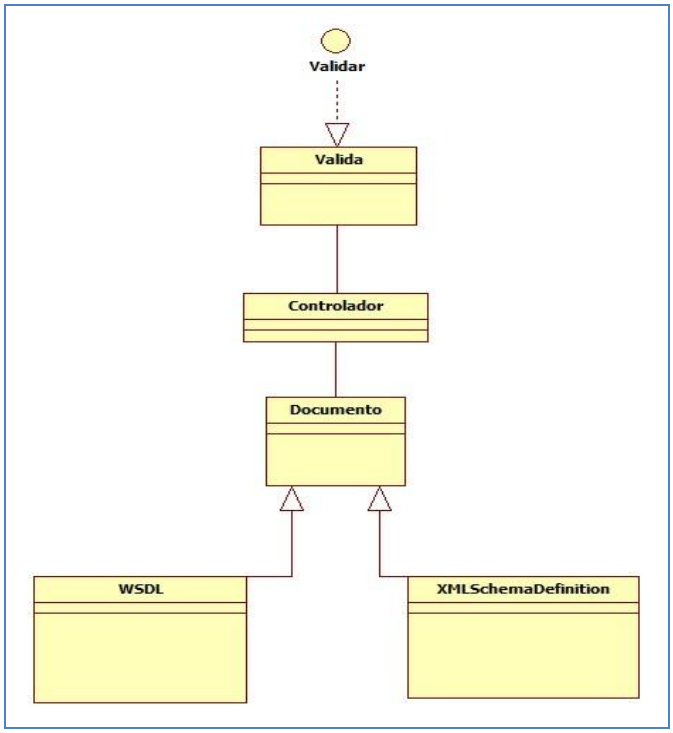
possuem o objetivo de fornecer uma interface simplificada para acesso as funcionalidades de uma biblioteca. Por essa razão, nas Seções 4.3.3 e 4.3.4, para realizar acesso às ontologias e Serviços Web disponíveis na web será utilizado o padrão de projeto *Facade*. Para facilitar o entendimento e ilustrar o uso desses padrões de projeto nos diagramas de classes construídos, cada classe no projeto que atenda as regras de implementação do *Business Logic Layer* levará no seu nome a sigla BLL, i.e.: <OwlBLL> e no mesmo sentido as classes no projeto que atendam as regras de implementação do *Data Access Layer* levarão nos seus nomes a sigla DAL, i.e.: <OwlDAL>. As classes que implementam o padrão de projeto Facade terão o nome <FacadeAcesso>.

Nas próximas Seções, são discutidas as implementações de cada serviço que faz parte do núcleo da anotação semântica, com base na representação dos respectivos diagramas de classes UML e suas funcionalidades.

4.4.1 *Structural Validation*

O diagrama de classe deste serviço é ilustrado na Figura 4-11 e possui uma interface denominada [Validar] na qual são definidas as assinaturas dos métodos que são disponibilizados através de um documento WSDL. A implementação dessa interface ocorre na classe [Validar] na qual são disponibilizadas algumas verificações iniciais (e.g.: formação do XML, existência de informação no documento, tamanho do arquivo) sobre os documentos recebidos.

Figura 4-11 Diagrama de classe para o serviço *Structural Validation*



Em seguida, os dados são recebidos pela classe [Controlador] que possuirá a responsabilidade de controlar o fluxo da informação através do qual ocorrerá a decisão em determinar qual tipo de documento será validado. A decisão é determinada através da técnica de Polimorfismo[92]. Nesse caso, a classe [Controlador] possui uma associação com a classe [Documento] no qual a depender do documento recebido será criado um objeto do tipo [WSDL] ou [XMLSchemaDefinition].

Nas classes [WSDL] e [XMLSchemaDefinition] ocorrem as implementações de acordo com o tipo de documento recebido. No caso desse documento ser o WSDL, são realizadas verificações na classe [WSDL] com o objetivo de validar a estrutura e regras definidas para os elementos de um documento WSDL 2.0, conforme detalhado no Capítulo 2. No caso do documento ser um arquivo XML *Schema Definition*, são consideradas apenas as verificações iniciais realizadas pela classe [Valida]. A classe foi criada com o objetivo de flexibilizar a inclusão de um comportamento novo (e.g.: surgimento de uma nova regra para criação de tipos complexos) para os documentos XML *Schema Definition*. Os detalhes sobre os documentos *XML Schema* também foram tratados no Capítulo 2.

[92] O Polimorfismo é uma técnica da programação orientação a objetos que possibilita as referências de tipo de classes mais abstratas representarem o comportamento das classes concretas que estão referenciando.

A utilização da técnica do Polimorfismo indica o grau de extensibilidade oferecido por este serviço, pois caso seja necessário validar um novo tipo de documento é necessário apenas criar uma nova classe que herde as características da classe [Documento] e implementar os comportamentos necessários. Além disso, o uso dessa técnica reduz o impacto para o cliente em uma manutenção evolutiva do serviço (*Structural Validation*).

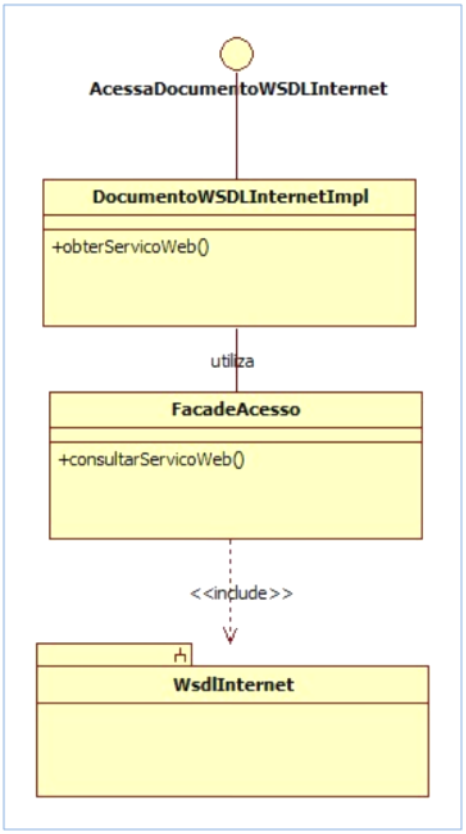
4.4.2 *Obtaining Web Services on the web*

Os comportamentos realizados por esse serviço são determinados pela interface [AcessaDocumentoWSDLInternet], cuja implementação ocorre na classe [AcessaDocumentoWSDLInternetImpl] (Figura 4-12).

Para realizar o acesso aos Serviços Web *restful* que estão disponibilizados na Internet o serviço *Semantic Annotation* utiliza o serviço *Obtaining Web Services on the web*. A classe [AcessaDocumentoWSDLInternetImpl] obtém a URI do *Web Service restful* a partir da classe [View] conforme discutido na Seção 4.4.5.

Na Figura 4-12 pode-se observar que a classe [FacadeAcesso] é responsável em estabelecer a comunicação na Internet para obtenção do WSDL a ser anotado semanticamente. Essa classe implementa o *design pattern Facade* cuja responsabilidade é fornecer a um cliente uma interface unificada para um conjunto de interfaces em um subsistema, sendo que nesse caso a classe [DocumentoWSDLInternetImpl] é o cliente, ocultando as complexidades do sistema e reduzindo a complexidade de comunicação entre subsistemas (GAMA; ERICH, 2000). Por essa razão, a classe [AcessaDocumentoWSDLInternetImpl] não precisa se preocupar com detalhes tais como conhecimento da API para acesso a documentos existentes na Internet, o que viabiliza o acesso no futuro a outros tipos de documentos que descrevam os serviços na web como, por exemplo, o WADL.

Figura 4-12 Diagrama de classe do serviço *Obtaining Web Services on the web*



A classe [WsdlInternet] representa o subsistema a ser acessado pela classe cliente e fornecerá o documento WSDL requisitado pelo usuário.

O documento WSDL 2.0 obtido da Internet pode ser utilizado pelo usuário do serviço para realizar a anotação semântica do *Web Service restful*, objeto de estudo deste trabalho, e posterior disponibilização em um repositório para favorecer a requisição, descoberta e composição automática desses Serviços Web.

4.4.3 *Obtaining Ontologies on the web*

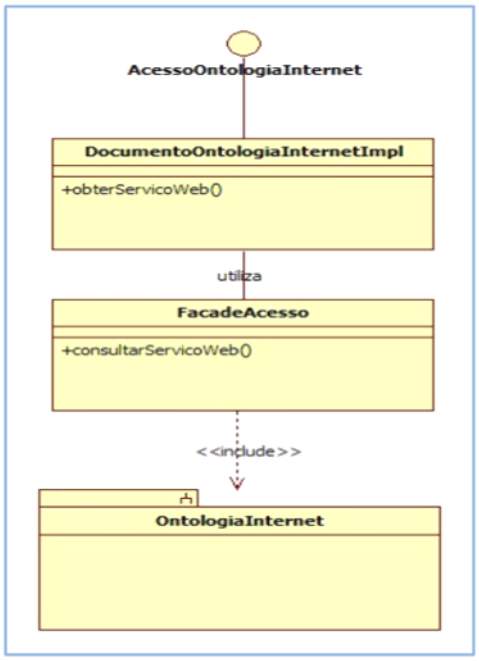
Este serviço, cujo diagrama de classe é ilustrado na Figura 4-13, possui uma interface denominada [AcessoOntologiaInternet] na qual são definidas as assinaturas dos métodos que viabilizarão obter as ontologias que estão localizadas na Internet.

Por essa razão, para realizar o acesso as ontologias que estão disponibilizados na Internet, o serviço *Semantic Annotation* utiliza o serviço *Obtaining Ontologies on the web*. A classe [DocumentoOntologiaInternetImpl] obtém a URI da ontologia OWL a partir da classe [View] conforme informado na Seção 4.3.2.

A classe [FacadeAcesso] da Figura 4-13 é responsável em estabelecer a comunicação na Internet para obtenção da ontologia OWL que terá por finalidade determinar o vocabulário

semântico do *Web Service restful*. Essa classe possui o mesmo objetivo da classe [FacadeAcesso] da Figura 4-12.

Figura 4-13 Diagrama de classe do serviço *Obtaining Ontologies on the web*



Para ter acesso as ontologias OWL disponibilizadas na Internet, a classe [DocumentoOntologiaInternetImpl] faz uso da classe [FacadeAcesso] no qual ocorrem as implementações para tal finalidade. A classe [OntologiaInternet] representa o subsistema a ser acessado pela classe [FacadeAcesso] e fornecerá a ontologia OWL requisitado pelo usuário.

O documento OWL obtido da Internet pode ser utilizado pelo usuário do serviço para determinar o vocabulário semântico do documento WSDL objeto de pesquisa desse trabalho.

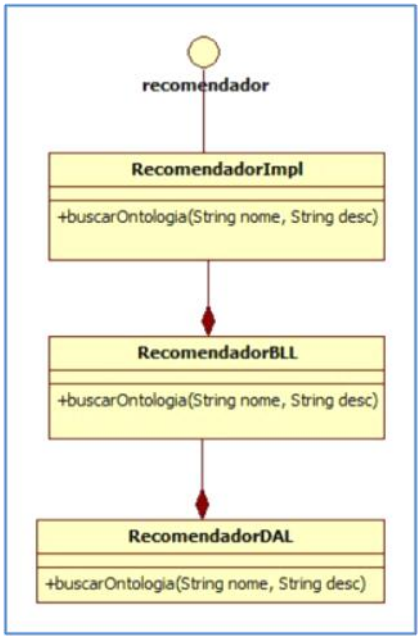
4.4.4 *Ontology Recommender*

Este serviço, cujo diagrama de classe é ilustrado na Figura 4-14, possui uma interface denominada [Recomendador] na qual são definidas as assinaturas dos métodos que viabilizarão ao SWS Editor fornecer ontologias para realizar anotação de semântica de Serviços Web similares aos já anotados semanticamente pela ferramenta.

Os comportamentos realizados por esse serviço são determinados pela interface [Recomendador], cuja implementação ocorre na classe [RecomendadorImpl] (Figura 4-14).

A classe [RecomendadorImpl] delega a classe [RecomendadorBLL] a execução da operação responsável em sugerir as ontologias para realizar as anotações semânticas nos Serviços Web. Essas ontologias estão armazenadas no repositório de ontologias e para acessá-las é realizada uma consulta em banco de dados através da classe [RecomendadorDAL].

Figura 4-14 Diagrama de classe do serviço *Ontology Recommender*



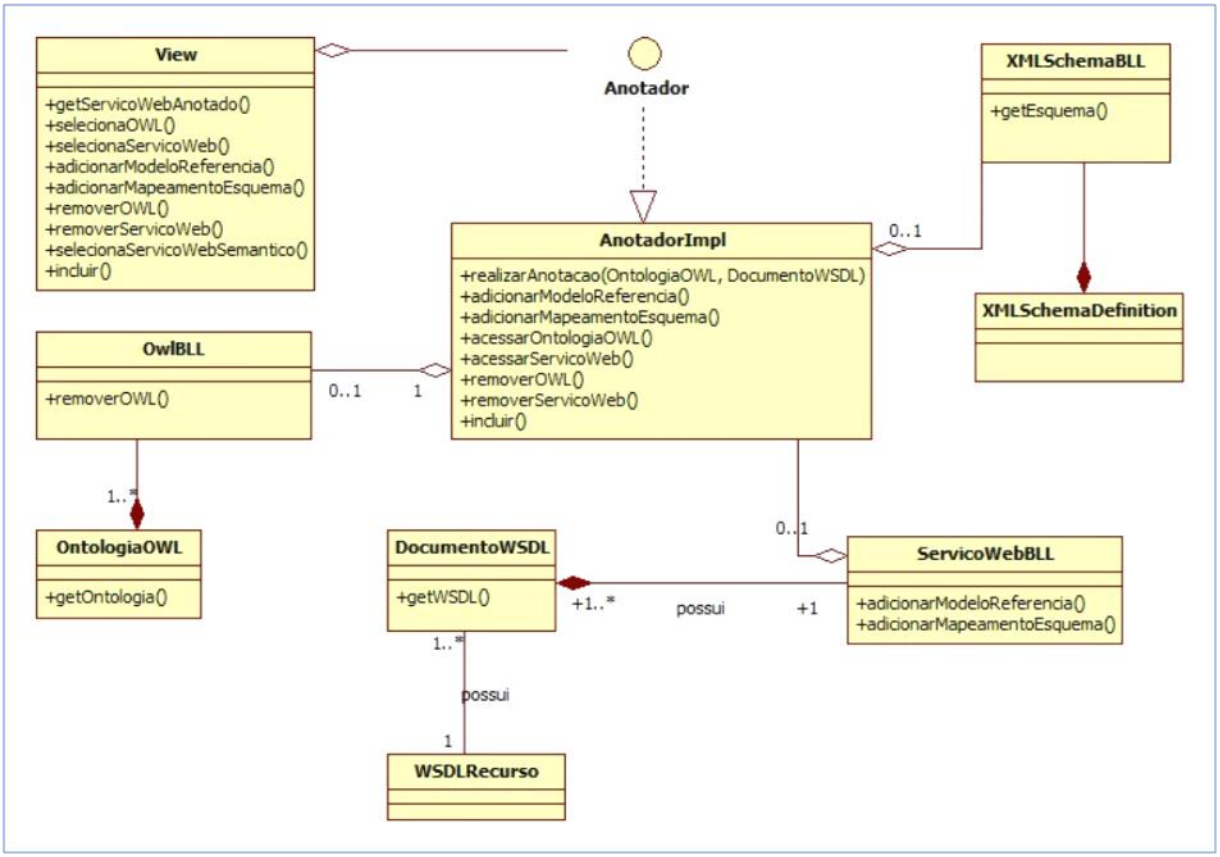
4.4.5 *Semantic Annotation*

Os comportamentos realizados por esse serviço são determinados pela interface [Anotador], cuja implementação ocorre na classe [AnotadorImpl], que executa as tarefas para o fornecimento da semântica aos documentos (objetos) WSDL 2.0 e XML *Schema Definition*.

Na Figura 4-15 pode-se observar o diagrama de classe que representa este serviço. A classe [AnotadorImpl] possui a capacidade de realizar as operações solicitadas pela classe [View] sendo responsável em comandar a anotação semântica dos documentos (WSDL e XML *Schema Definition*) a partir da especificação SAWSDL que foi discutido no Capítulo 2. Os procedimentos para realizar a anotação semântica desses documentos foram discutidos na Seção 4.3.5.

A classe [AnotadorImpl] executa operações que são delegados às classes [OwlBLL]**,** [ServicoWebBLL] e [XMLSchemaBLL] respectivamente. Essas operações estão relacionadas aos comportamentos específicos para utilização da ontologia OWL, documento WSDL e documento XML *Schema Definition.*

Figura 4-15 Diagrama de Classe para o serviço *Semantic Annotation*



A classe [View] representa a camada de visão da ferramenta e possui a responsabilidade de fornecer a interação com o usuário disponibilizando os eventos que darão início a anotação semântica dos Serviços Web *restful*. Essa classe ainda possui como responsabilidade obter do usuário a URL dos Serviços Web *restful* e ontologias OWL que estão disponibilizados na Internet. A implementação relacionada ao acesso aos Serviços Web *restful* e ontologias OWL que estão disponibilizados na Internet foi explicada nas Seções 4.4.2 e 4.4.3.

A classes [DocumentoWSDL], [XMLSchemaDefinition] e [OntologiaOWL] possuem a responsabilidade de representar os respectivos documentos XML em memória, ou seja, os documentos que descrevem sintaticamente os serviços *web restful* e os documentos que determinam o vocabulário semântico desses serviços que são transformados em objetos na ferramenta. A classe [WSDLRecurso] tem como finalidade a caracterização dos recursos e expressa as capacidades de um serviço.

Ao final da anotação semântica de um documento WSDL é fornecido ao usuário da aplicação o documento anotado semanticamente a partir da especificação SAWSDL (e.g.: Figura 4-16).

Figura 4-16 Trecho de um documento WSDL anotado semanticamente

```
12    ...
13   <wsdl:interface name="BookListInterface" sawsdl:modelReference="http://example.org/categorization/books/technologies">
14    <wsdl:operation name="getBookList"   sawsdl:modelReference="http://example.org/spec/ontology/bookList#RequestbookList"
15      pattern="http://www.w3.org/ns/wsdl/in-out"
16      style="http://www.w3.org/ns/wsdl/style/iri"
17      wsdlx:safe="true">
18      <wsdl:documentation>
19        This operation returns a list of books.
20      </wsdl:documentation>
21      <wsdl:input element="msg:getBookList"/>
22      <wsdl:output element="msg:bookList"/>
23    </wsdl:operation>
24   </wsdl:interface>
25
26   <wsdl:binding name="BookListHTTPBinding"
27     type="http://www.w3.org/ns/wsdl/http"
28     interface="tns:BookListInterface">
29    <wsdl:documentation>
30      The RESTful HTTP binding for the book list service.
31    </wsdl:documentation>
32    <wsdl:operation ref="tns:getBookList" whttp:method="GET" sawsdl:modelReference="http://example.org/spec/ontology/GetbookList#RequestGetbookList"/>
33   </wsdl:binding>
34   ...
35  </wsdl:description>
```

O documento gerado contém o atributo *modelReference* (explicado no Capítulo 2) sendo utilizados nas linhas 13, 14, e 32 para anotar semanticamente os elementos *interface* e *operation* do WSDL.

No caso do *modelReference* utilizado na linha 13, para realizar a anotação semântica do elemento *interface*, aponta para uma categoria de conceito "*tecnologies*" no modelo semântico determinado pela URI no qual é responsável em descrever e determinar os aspectos comportamentais desse elemento (e.g. determina o que o serviço faz). Nas linhas 14 e 32, o *modelReference* é utilizado para anotar o elemento *operation* com o objetivo de fornecer a descrição do funcionamento do serviço e determinar o comportamento oferecido pela operação (nesse exemplo, o comportamento oferecido pela operação getBookList, a partir de uma invocação GET do protocolo HTTP, será o retorno de um conjunto de livros). Esse funcionamento e comportamento é modelado pela ontologia determinada na URI do atributo *modelReference*.

Esse documento de saída, contendo anotações semânticas nos elementos de *interface* e *operation*, pode ser disponibilizado pelo usuário da ferramenta em qualquer repositório que possua a finalidade de favorecer, por exemplo, a descoberta automática de serviços.

A Figura 4-17 ilustra a anotação semântica realizada em um documento XML *Schema Definition*. O trecho do documento possui o atributo de extensão *liftingSchemaMapping*

(explicado no Capítulo 2) sendo utilizado na linha 24 para anotar semanticamente o elemento complexType. Nesse caso, o XSLT (discutido no Capítulo 2) é utilizado como linguagem para especificar o mapeamento dos elementos contidos no *XML Schema Definition* para os conceitos definidos no modelo semântico.

Para realizar o mapeamento dos dados contidos em um arquivo XSLT, uma ferramenta cliente recupera o valor fornecido pela URI no atributo (sawsdl:liftingSchemaMapping) e aplica aos elementos especificados pelo *XML Schema Definition.* Na Figura 4-17, o arquivo XSLT determinado pela URI http://example.org/mapping/BookList.xslt será aplicado aos elementos "*author*", "*title*", "*publisher*", "*subject*" e "*language*".

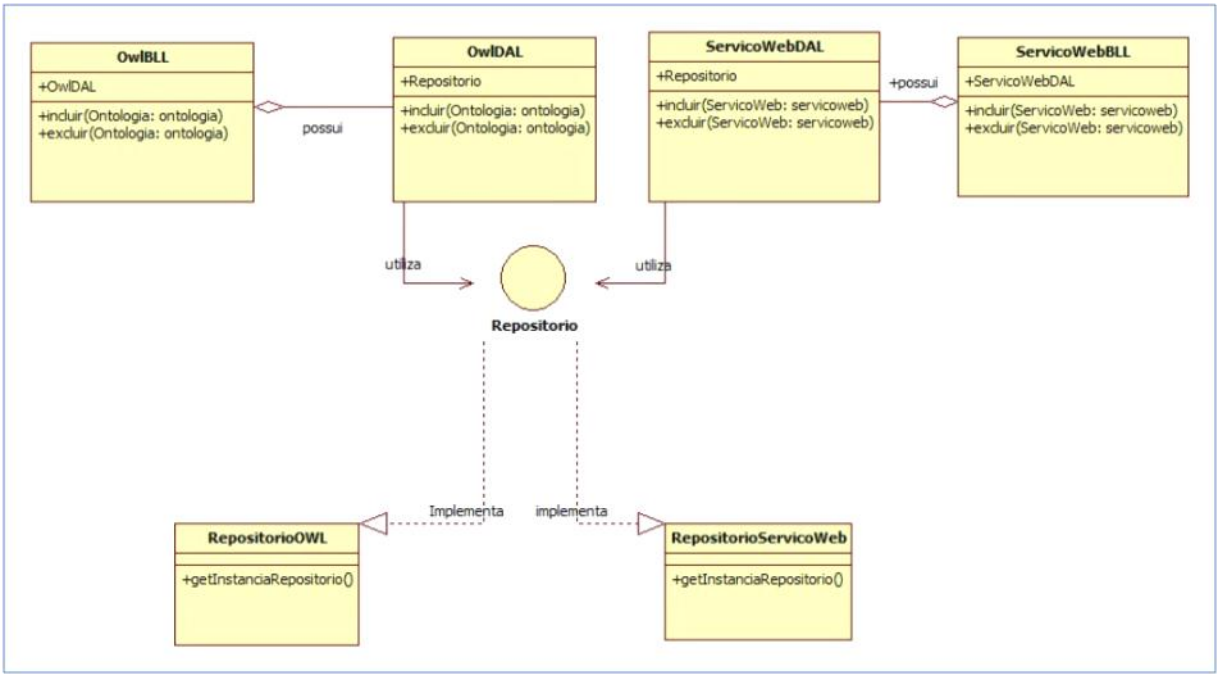
Figura 4-17 Trecho de um documento XML Schema Definition anotado semanticamente

```
23  ...
24   <complexType name="getBookListType" sawsdl:liftingSchemaMapping="http://example.org/mapping/BookList.xslt">
25    <sequence>
26     <element name="author" type="string" minOccurs="0" maxOccurs="unbounded"/>
27     <element name="title" type="string" minOccurs="0" maxOccurs="1"/>
28     <element name="publisher" type="string" minOccurs="0" maxOccurs="1"/>
29     <element name="subject" type="string" minOccurs="0" maxOccurs="1"/>
30     <element name="language" type="string" minOccurs="0" maxOccurs="unbounded"/>
31    </sequence>
32   </complexType>
33  ...
```

4.4.6 Implementação de classes para suporte aos Serviços

Para o funcionamento dos serviços algumas funcionalidades necessitaram ser implementadas: (i) armazenamento dos Serviços Web *restful* e o (ii) armazenamento de ontologias OWL. Para ilustrar a modelagem dessas funcionalidades foram elaborados diagramas de sequência e de classe. O diagrama de classe apresentado na Figura 4-19 representa a estrutura para o armazenamento dos Serviços Web e das ontologias OWL.

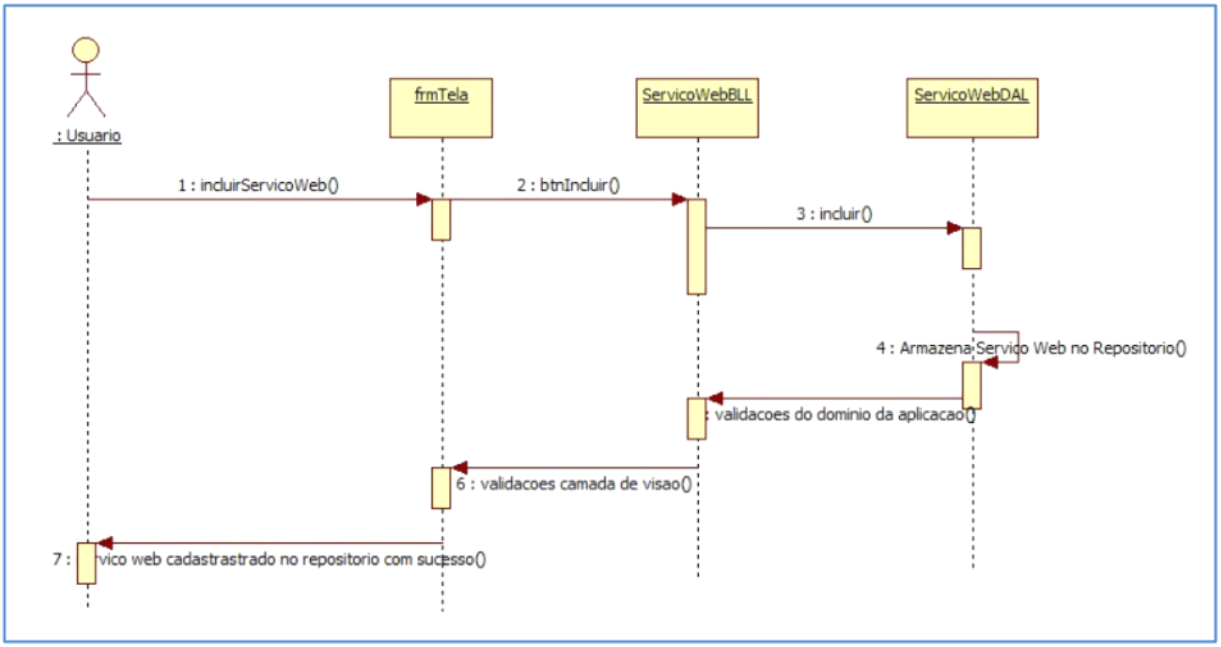
Figura 4-18 Diagrama de classes para o armazenamento das Ontologias e Serviços Web



Esse diagrama (Figura 4-18) define uma interface que representa o [Repositorio] e possui sua implementação definida nas classes de implementação [RepositorioOWL] e [RepositorioServicoWeb]. Essa especialização possibilita, caso haja necessidade, tratar de forma diferenciada a tecnologia de armazenamento para as ontologias e Serviços Web (i.e.: podem ser utilizados dois sistemas de banco de dados diferentes). As classes [OwlBLL] e [ServicoWebBLL] serão utilizadas para a realização de regras negócios que façam parte do domínio do problema. As classes [OWLDAL] e [ServicoWebDAL] possuem como responsabilidade e capacidade estabelecer as instruções que armazenarão ou excluirão as informações do Repositório.

O diagrama de sequência, ilustrado na Figura 4-19 representa a sequência de processos necessários para o armazenamento de Serviços Web *restful*. Nessa etapa, o documento WSDL 2.0 é armazenado em um repositório local.

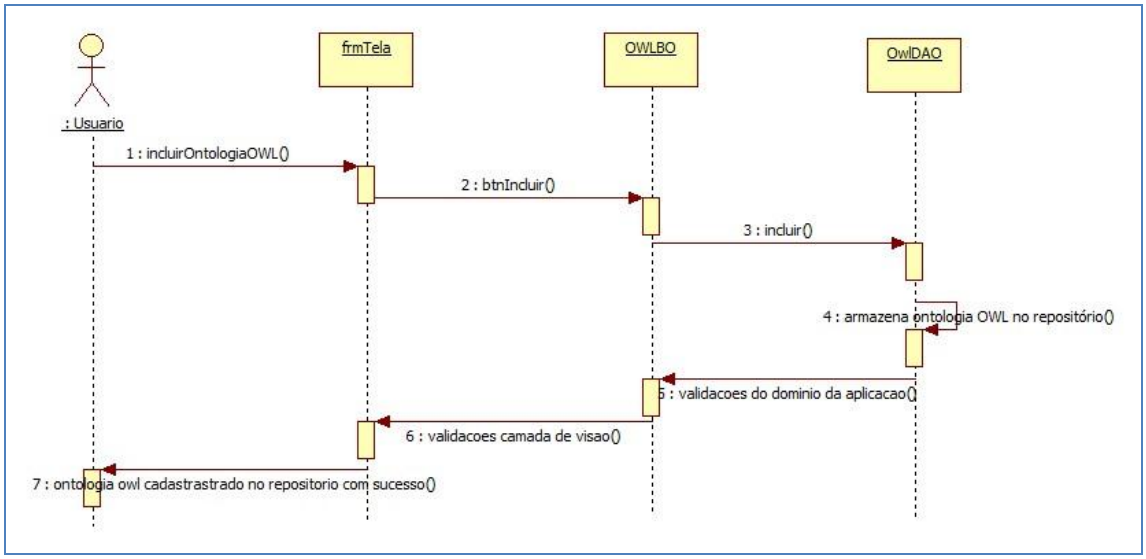
Figura 4-19 Diagrama de sequência para o armazenamento de Serviços Web *restful*



A funcionalidade representada nesse diagrama (Figura 4-19) manterá o controle de todos os Serviços Web que precisam ser anotados e os Serviços Web já anotados semanticamente. O usuário da ferramenta SWS Editor solicita a inclusão do Serviços Web que poderá estar ou não anotado semanticamente. A SWS Editor poderá realizar validações que estejam relacionadas ao domínio da aplicação para em seguida efetuar o armazenamento dos Serviços Web *restful* ou Serviços Web *restful* semânticos. Por fim, uma mensagem é informada ao usuário da inclusão realizada com sucesso do *Web Service restful* no Repositório.

O diagrama de sequência (Figura 4-20) representa a sequência de atividades necessárias para o armazenamento das ontologias OWL. A representação dessas atividades no diagrama de sequência é similar a representação ilustrada para o armazenamento de Serviços Web *restful*. O usuário solicita a inclusão da ontologia OWL que será utilizada para determinar o vocabulário semântico dos Serviços Web *restful*. A ferramenta SWS Editor realizará validações que estejam relacionadas ao domínio da aplicação (e.g. não é possível realizar a inclusão de ontologia já existente no repositório) para em seguida efetuar o armazenamento da ontologia. Por fim, uma mensagem é informada ao usuário da inclusão da ontologia no repositório.

Figura 4-20 Diagrama de sequência para o armazenamento de ontologias OWL



4.5 CONSIDERAÇÕES FINAIS

Neste capítulo foi apresentada a ferramenta SWS Editor, que permite a anotação semântica em Serviços Web *restful*. Sua implementação foi baseada em serviços visando a interoperabilidade e reuso da ferramenta SWS Editor em diferentes ambientes.

Quadro 4-1 Avaliação dos trabalhos

	R1	R2	R3	R4	R5	R6	R7	R8	R9
SWS Editor	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim	Sim
Anotação Semântica	Não	Não	Não	Sim	Sim	Sim	Sim	Não	Sim
SWEET	Sim	Não	Não	Parcialmente	Não	Não	Não	Não	Sim
WSMO Studio	Não	Não	Não	Parcialmente	Não	Sim	Sim	Não	Sim
RADIANTWEB	Sim	Não	Sim	Parcialmente	Sim	Não	Não	Não	Sim
IRIDESCENT	Sim	Sim	Sim	Sim	Sim	Não	Não	Não	Sim

A comparação das ferramentas discutidas no Capítulo 3 com a ferramenta SWS Editor, proposta nesta dissertação, é ilustrada no Quadro 4-1, no qual é possível observar o atendimento aos requisitos considerados essenciais para a anotação semântica, discutidos no Capítulo 3. Os requisitos (R1) anotação semântica de Serviços Web *restful* descritos com

WSDL 2.0, (R2) anotação de Serviços Web *restful* e (R9) anotação de serviços no padrão SOAP são atendidos na ferramenta SWS Editor a partir da implementação dos serviços *Structural Validation*, *Obtaining Web Services on the web e Semantic Annotation* discutidos nas Seções 4.3.1, 4.3.2, 4.3.5, 4.4.1, 4.4.2 e 4.4.5. O requisito (R3) Utilização de ontologias disponibilizadas na web é atendido pela ferramenta a partir da implementação do serviço *Obtaining Ontologies on the web* discutido nas Seções 4.3.3 e 4.4.3. Com relação ao atendimento do requisito (R4) Utilização de tecnologias que são recomendações do W3C, as tecnologias (e.g.: XML, WSDL, RDF, OWL, SAWSDL) discutidas no Capítulo 2 e que subsidiam a construção da ferramenta SWS Editor são recomendadas pelo W3C. Por essa razão todos os serviços implementados neste trabalho fazem uso dessas tecnologias. O requisito (R5) Utilização de conceitos ontológicos definidos em OWL são atendidos na Ferramenta SWS Editor a partir da implementação dos serviços *Obtaining Ontologies on the Web* e *Ontology Recommender* discutidos nas Seções 4.3.3, 4.3.4, 4.4.3 e 4.4.4. O vocabulário semântico determinado nos documentos WSDL e *XML Schema Definition* é realizado através de ontologias construídas em linguagem OWL (As ontologias construídas em linguagem OWL e utilizadas neste trabalho podem ser verificadas em um repositório na web: (https://github.com/cleberlira/SWS-Editor/tree/master/Ontologias). Os requisitos (R6) manter um repositório de *Web Services* Semânticos e (R7) Manter um repositório de ontologias são implementados respectivamente pelos serviços *Obtaining Web Services on the web*, *Obtaining Ontologies on the web* e *Semantic Annotation* que armazenam os respectivos documentos WSDL e OWL em um banco de dados (modelagem disponível no apêndice B). O requisito (R8) que trata da extensibilidade da ferramenta SWS Editor e da sua implementação a partir do conceito de serviços que podem ser verificados também no repositório disponibilizado na web: (https://github.com/cleberlira/SWS-Editor/tree/master/Serviços) onde estão localizados os documentos WSDL referente a cada serviço implementado. O Quadro 4-2 mostra uma matriz de rastreabilidade relacionando os serviços oferecidos pela ferramenta SWS Editor e os requisitos atendidos por esses serviços. No Quadro 4-2, a letra X indica que o serviço implementa o requisito especificado.

Quadro 4-2 Matriz de rastreabilidade Serviços X Requisitos

	(S1) *Structural Validation*	(S2) *Semantic Annotation*	(S3) *Obtaining Web Services on the Web*	(S4) *Obtaining Ontologies on the web*	(S5) *Ontology Recommender*
R1	X	X			
R2	X	X			
R3				X	
R4	X	X	X	X	X
R5				X	X
R6		X	X		
R7		X		X	
R8	X	X	X	X	X
R9	X	X			

O Capítulo 5 discute um exemplo de uso da ferramenta através do qual um documento WSDL 2.0 que descreve sintaticamente um *Web Service restful* é anotado semanticamente. Neste capítulo é exemplificado o acesso a cada serviço discutido nas Seções 4.3 e 4.4 e o atendimento aos requisitos apresentados no Capítulo 3.

5 EXEMPLO DE USO DA FERRAMENTA SWS Editor – PROVA DE CONCEITO

Este capítulo destina-se a exemplificar o uso da ferramenta SWS Editor para ilustrar o acesso aos serviços e realizar a anotação semântica de Serviços Web *restful.* O ambiente de desenvolvimento do exemplo de uso inclui aplicações clientes que acessam os serviços: (i) *Structural Validation*; (ii) *Obtaining Web Services on the web;* (iii) *Obtaining Ontologies on the web;* (iv) *Ontology Recommender* e, por fim, (v) realização da anotação semântica de um documento WSDL 2.0 descrevendo sintaticamente um *Web Service restful*, além da inserção e requisição do *Web Service restful* no repositório semântico.

O ambiente para aplicação do exemplo de uso foi preparado com a instalação de aplicações clientes do tipo *windows forms*[93] em diferentes máquinas com o objetivo de consumir os serviços disponibilizados em um servidor web, contendo a infraestrutura necessária discutida na Seção 4.4. A análise dos resultados desses testes ocorreu a partir da observação do comportamento da *SWS Editor* durante o consumo dos serviços e adequação ao modelo de funcionamento proposto na Seção 4.3.

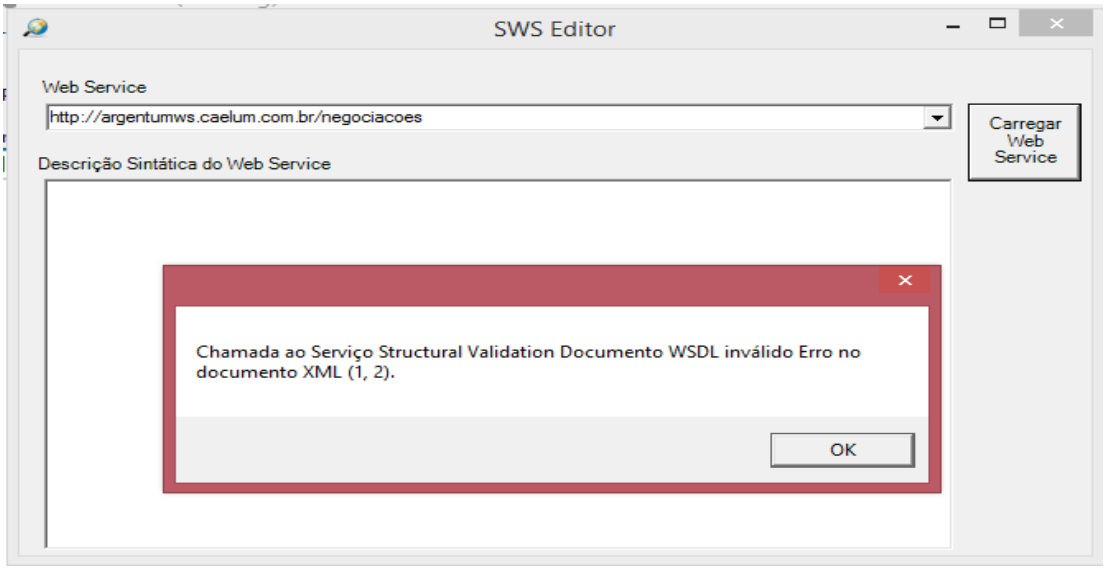
Os testes aplicados foram definidos de acordo com os seguintes cenários de consumo:

a) acesso ao serviço *Structural Validation* com um documento que não atenda às regras de descrição para um documento WSDL (e.g.: utilização de *namespaces*, elementos e atributos inapropriados). A Figura 5-1 ilustra o acesso ao *Web Service* (http://argentumws.caelum.com.br/negociacoes)[94] que não está descrito sintaticamente em um documento WSDL.

[93] https://msdn.microsoft.com/pt-br/library/dd30h2yb(v=vs.110).aspx

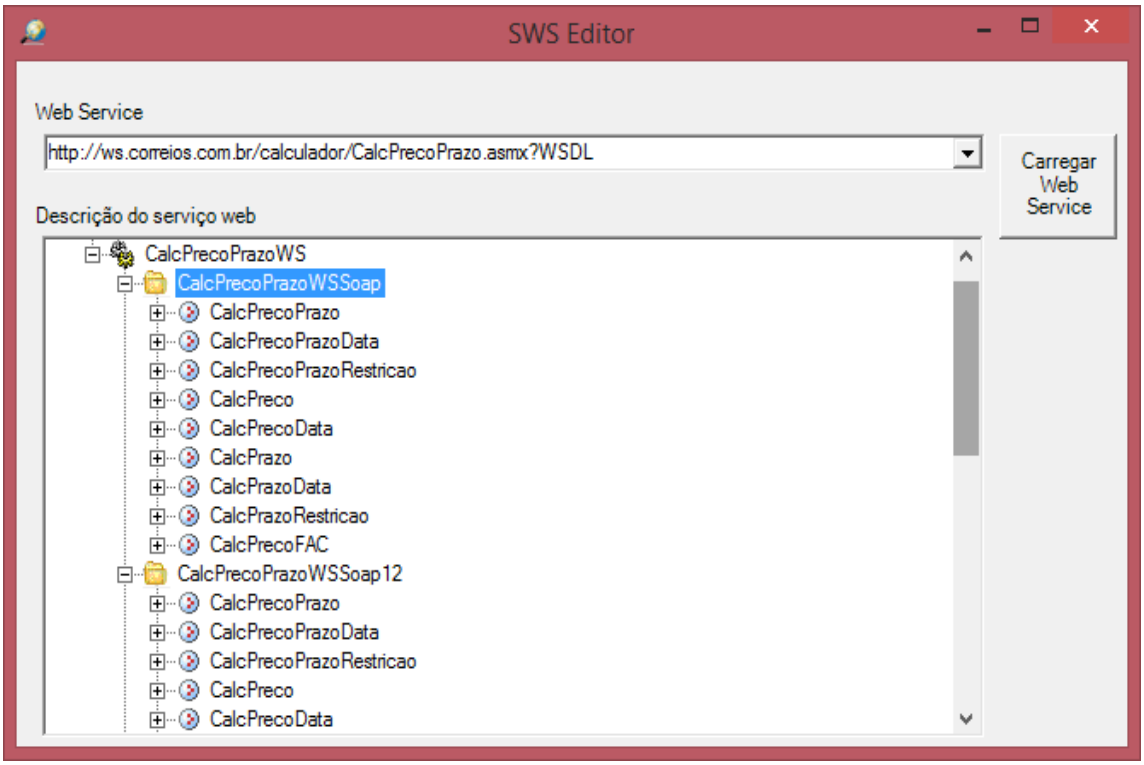
[94] http://www.caelum.com.br/apostila-java-testes-jsf-web-services-design-patterns/acessando-um-web-service/#6-2-consumindo-dados-de-um-web-service

Figura 5-1 Exemplificação de acesso ao serviço *Structural Validation*



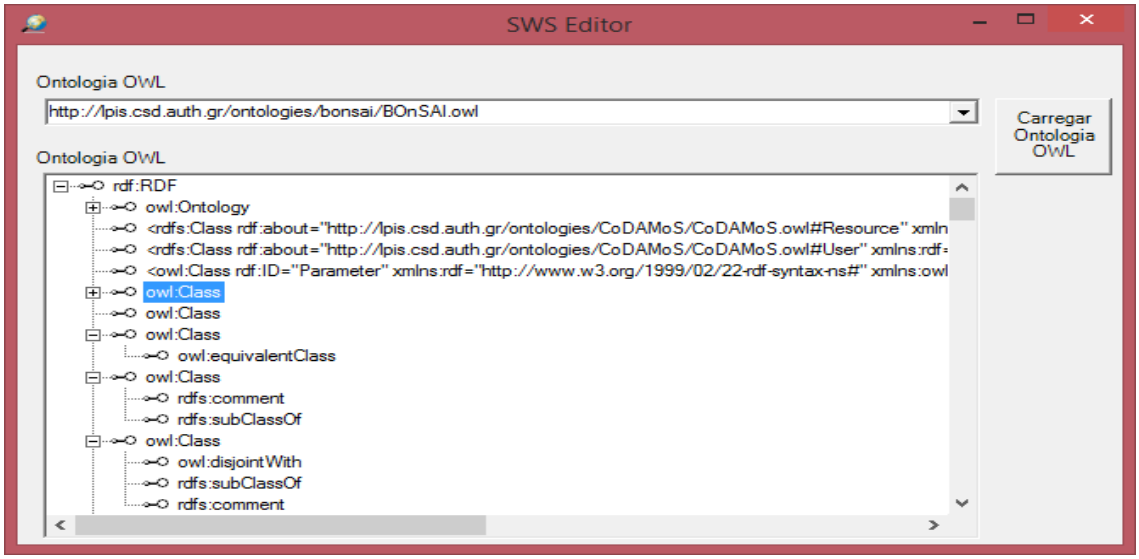
b) acesso ao serviço Obtaining Web Services on the web a partir da informação da URL do Web Service pelo usuário. O funcionamento e implementação deste serviço foram discutidos nas Seções 4.3.2 e 4.4.2 respectivamente. A Figura 5-2 ilustra o acesso ao Web Service (http://ws.correios.com.br/calculador/CalcPrecoPrazo.asmx?WSDL).

Figura 5-2 Exemplificação de acesso ao serviço *Obtaining Web Services on the web*



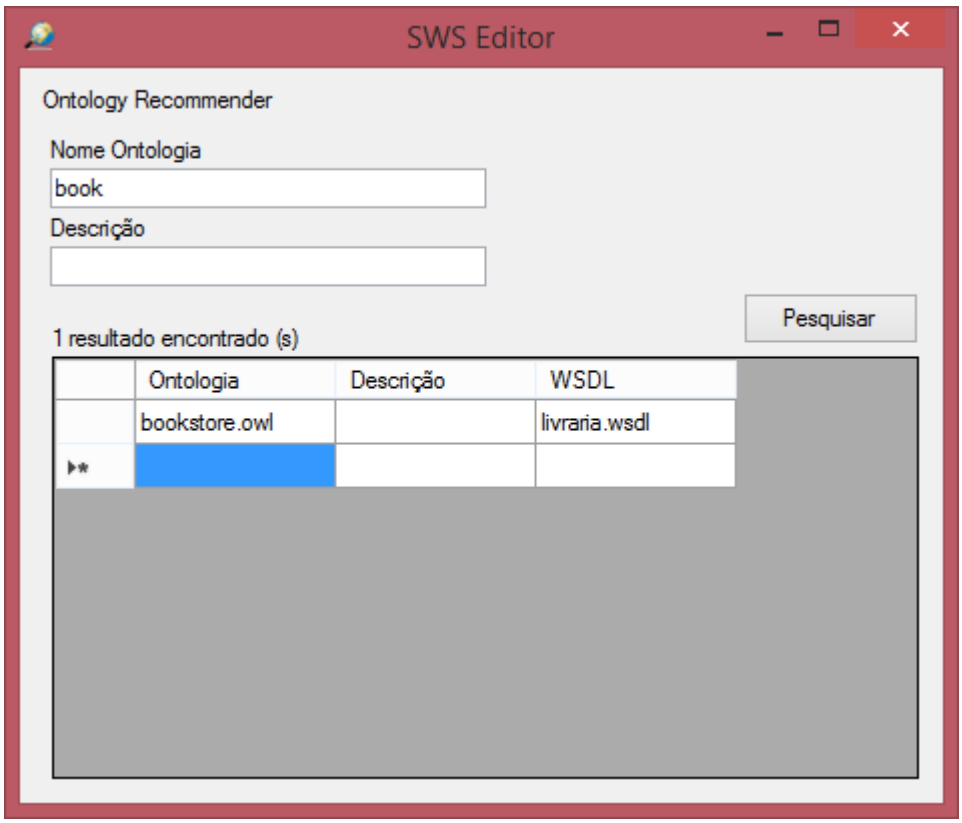
c) acesso ao serviço Obtaining Ontologies on the web a partir da informação da URL da ontologia OWL pelo usuário. O funcionamento e implementação desse serviço foram discutidos nas Seções 4.3.3 e 4.4.3 respectivamente. A Figura 5-3 ilustra o acesso a ontologia OWL disponível em (http://lpis.csd.auth.gr/ontologies/bonsai/ BOnSAI.owl).

Figura 5-3 Exemplificação de acesso ao serviço *Obtaining Ontologies on the web*



d) acesso ao serviço Ontology Recommender (Figura 5-4), nesse exemplo, o processamento é iniciado a partir do fornecimento da string que contém o nome da ontologia. O SWS Editor realiza a busca a partir de três caracteres informado. O usuário ao clicar no botão Pesquisar aciona o serviço Ontology Recommender que retorna como resposta a quantidade de ontologias encontradas, a lista das ontologias existentes no sistema e que já foram utilizadas para fornecer o vocabulário semântico de documentos WSDL.

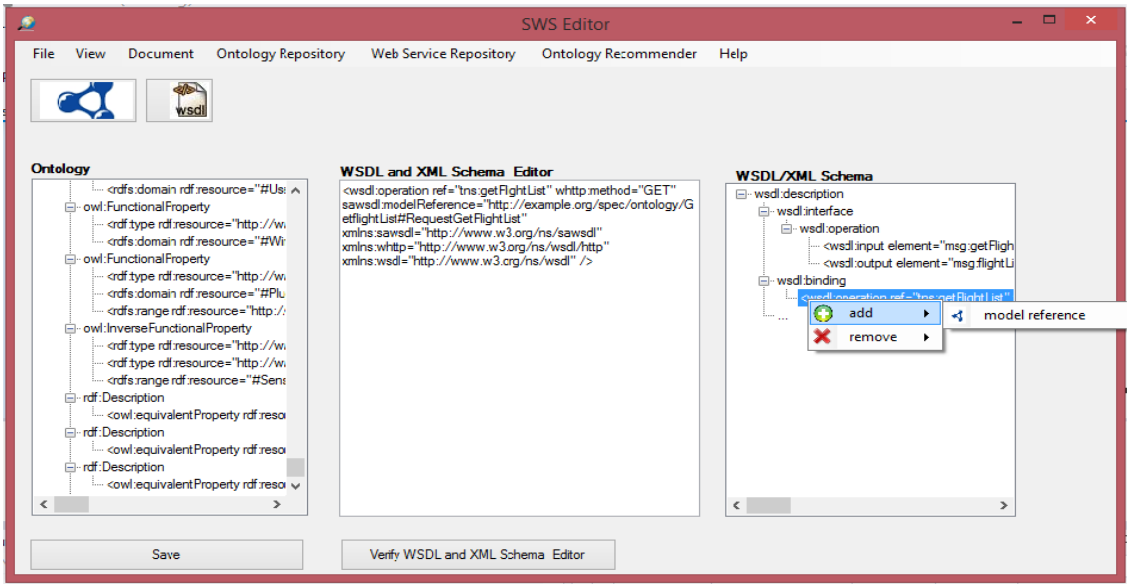
Figura 5-4 Exemplificação de acesso ao serviço *Ontology Recommender*



e) (acesso ao serviço Semantic Annotation.

Para realizar a anotação semântica dos Serviços Web *restful* a SWS Editor, conforme ilustrada na Figura 5-5, é composta por três componentes principais: (i) *Ontology*, cuja responsabilidade é o carregamento da ontologia que irá descrever o *Web Service*; (ii) WSDL and XML *Schema Editor*, cujo objetivo é permitir ao usuário a edição dos documentos a serem anotados semanticamente (a anotação semântica pode ser realizada também utilizando a opção de *context menu*, acionada ao clicar com o botão direito do mouse e selecionar o elemento do documento WSDL ou XML *Schema Definition*) e o (iii) WSDL/XML Schema, cuja responsabilidade é realizar a carga do documento WSDL ou XML *Schema Definition*. O menu da SWS Editor oferece acesso a outras opções como: *Ontology Repository*, permite ao usuário o acesso as ontologias armazenadas, possibilitando também a inclusão e remoção desses documentos. Esse repositório será disponibilizado na web por meio de Serviços Web; o *Web Service Repository* possibilita aos usuários consulta aos Serviços Web descritos sintaticamente e aos Serviços Web semânticos e possibilita a remoção e inclusão de novo serviços. Esse repositório também poderá estar disponível na web por meio de Serviços Web.

Figura 5-5 Tela Principal do SWS Editor



A Listagem 5-1 apresenta o trecho da descrição de um *Web Service restful* descrito sintaticamente em WSDL e anotado semanticamente com SAWSDL utilizando a ferramenta proposta nesta dissertação, e.g. SWS Editor. A linha 5 ilustra o uso do *modelReference* para categorizar o elemento interface *BookListInterface*, enquanto a linha 7 associa a operação getBookList com a classe *bookList*. Da mesma forma, a linha 10 ilustra o *modelReference* através da classe *GetbookList* para determinar o comportamento da operação *getBookList* ao ser invocado a partir da operação GET.

Listagem 5-1: Trecho de um *Web Service restful* anotado com SAWSDL utilizando a ferramenta SWS Editor

---

1. <?xml version="1.0" encoding="UTF-8"?>

2.<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"

3...

4. <wsdl:interface name="BookListInterface"

5.sawsdl:modelReference="http://example.edu.unifacs/ontologies/bookstore.owl# Technology">

6. <wsdl:operation name="getBookList"

7.sawsdl:modelReference=" http://example.edu.unifacs/ontologies/bookstore.owl#BookList"

8. pattern= style="http://www.w3.org/ns/wsdl/style/iri" wsdlx:safe="true">

9. <wsdl:input element="msg:getBookList"/>

10. <wsdl:output element="msg:bookList"/>

11. </wsdl:operation>

12.</wsdl:interface>

13.<wsdl:binding name="BookListHTTPBinding" type=http://www.w3.org/ns/wsdl/http

14. interface="tns:BookListInterface">

15. <wsdl:documentation>

16. The RESTful HTTP binding for the book list service.

17. </wsdl:documentation>

18. <wsdl:operation ref="tns:getBookList" whttp:method="GET"
19.sawsdl:modelReference=" http://example.edu.unifacs/ontologies/bookstore.owl#RequestGetbookList"/>

20. </wsdl:binding>
21....
22.</wsdl:description>

Para fornecer a semântica do *Web Service restful* ilustrado na Listagem 5-1 foi construída a ontologia bookstore.owl [95] através da ferramenta Protégé [96]. Esta ontologia descreve a representação dos livros de tecnologia através da classe ontológica Technology. A linha 5, na Listagem 5-1, o *modelReference* aponta para a classe ontológica Technology através da IRI http://example.edu.unifacs/ontologies/bookstore.owl#Technology. A linha 7 aponta para a classe ontológica [BookList] através da IRI http://example.edu.unifacs/ontologies/bookstore.owl#BookList que é responsável em fornecer a descrição semântica da operação <getBookList> especificando aspectos comportamentais (e.g.: tipo de retorno da operação). A linha 19 aponta para a classe ontológica [RequestGetbookList] para determinar o comportamento da operação <getBookList> ao ser invocado a partir da operação GET.

Para exemplificar a definição do repositório semântico, conforme discutido no Capítulo 2, a partir do armazenamento das anotações semânticas localizadas no documento WSDL e adequação ao requisito (R6): Manter um repositório de Serviços Web Semânticos foram realizadas implementações conforme ilustradas nas Listagens 5-2, 5-3 e 5-4. De acordo com Boiissel-Dallier, Lorré e Benaben (2009) as anotações semânticas podem ser retiradas do documento WSDL através da API *EasySAWSDL*[97]. Essa API percorre todos os elementos de

[95] https://github.com/cleberlira/SWS-Editor/blob/master/Ontologias/bookstore.owl

[96] Este trabalho foi realizado utilizando o recurso Protégé, que é apoiado pela concessão GM10331601 do Instituto Nacional de Ciências Médicas Gerais dos Estados Unidos National Institutes of Health.

[97] http://easywsdl.ow2.org/extensions-sawsdl.html

um documento WSDL em busca das anotações que se referem a algum conceito ontológico. As Listagens 5-2 e 5-3 mostram exemplos de inclusões no repositório semântico utilizando a linguagem SPARQL UPDATE através do *framework Jena*, ambos discutidos no Capítulo 2. Essas informações mantidas em um repositório podem ser utilizadas posteriormente para realizar a descoberta e composição automática de serviços. Nessas Listagens são incluídos os livros na categoria determinada pela IRI: http://example.edu.unifacs/categorization/books. O APENDICE E contém o código completo relacionado as Listagens 5-2, 5-3 e 5-4.

Listagem 5-2 Exemplo de inclusão no repositório semântico

```
1.PREFIX sawsdl: <http://example.edu.unifacs/ontologies/bookstore.owl#Technology >
2.INSERT DATA{
3.<http://example.edu.unifacs/categorization/books> sawsdl: 'Java'
4.}
```

Listagem 5-3 Exemplo de inclusão no repositório semântico

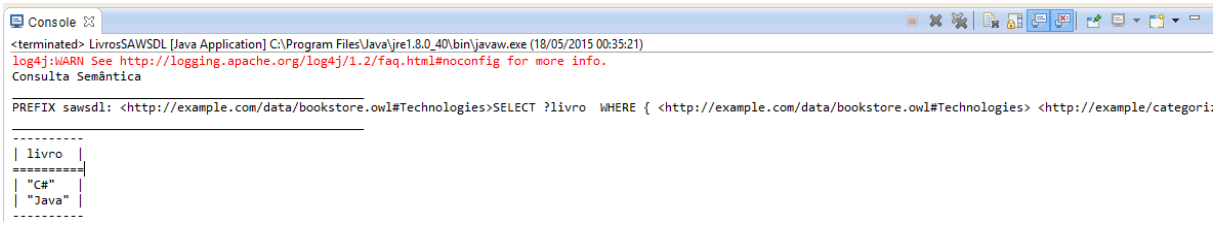
```
1.PREFIX sawsdl: <http://example.edu.unifacs/ontologies/bookstore.owl#Technology>
2. INSERT DATA{
3.<http://example/categorization/books> sawsdl: 'C#'
4.}
```

A Listagem 5-4 mostra um exemplo de como realizar uma consulta por serviços. Nesse caso a consulta retorna todos os livros que estão na categoria tecnologia. A Figura 5-6 ilustra o resultado da execução da consulta mostrada na Listagem 5-4.

Listagem 5-4 Exemplo de consulta semântica no repositório

```
1.PREFIX sawsdl: <http://example.edu.unifacs/ontologies/bookstore.owl#Technology >
2.SELECT ?livro  WHERE {
3.  <http://  example.edu.unifacs/ontologies/bookstore.owl#Technology  >
4.<http://example/categorization/books> ?livro .
5.}
```

Figura 5-6 Resultado da Consulta da Listagem 5-4



5.2 CONSIDERAÇÕES FINAIS

Os testes caixa preta realizados alcançaram o objetivo de ilustrar como se realiza a anotação semântica em Serviços Web por intermédio da ferramenta SWS – Editor, apresentada no Capítulo 4. De acordo com a expectativa das informações que devem ser retornadas por cada serviço, a exemplificação de uso da ferramenta mostrou que os resultados com o consumo dos serviços resultaram em respostas esperadas, ou seja, para cada solicitação a ferramenta apresenta respostas de acordo com os requisitos especificados (e.g.: anotação semântica de documentos WSDL 2.0, obtenção de Serviços Web e ontologias disponíveis na web, recomendação de ontologias) e discutidos no Capítulo 3.

Com relação a utilização da inclusão e recuperação de informações relacionada ao *Web Service restful* semântico baseado em SAWSDL foi ilustrada a sua aplicabilidade através de instruções SPARQL no *framework Jena*.

O exemplo de uso possibilitou confirmar a aplicabilidade da ferramenta SWS Editor, através da sua arquitetura, funcionalidades e serviços disponíveis que retornaram resultados satisfatórios nos cenários (e.g.: obtenção de ontologia na web, obtenção de Serviços Web na web, entre outros) a que foram aplicados.

6 CONCLUSÃO

Este capítulo descreve as conclusões e considerações finais sobre esta dissertação, suas principais contribuições e possibilidades de trabalhos futuros. Ele está organizado da seguinte maneira: inicialmente, na Seção 6.1 são discutidas as considerações finais sobre o trabalho; a Seção 6.2 discute as principais contribuições; a Seção 6.3 informa sobre os artigos publicados durante a elaboração desta dissertação; e a Seção 6.4 apresenta as possibilidades de trabalhos futuros.

6.1 CONSIDERAÇÕES FINAIS

A necessidade de realizar anotações semânticas em Serviços Web *restful* foi considerada nesta dissertação, visto que atualmente aplicações da Web 2.0 disponibilizam suas interfaces através desses serviços, entretanto, essas aplicações não possibilitam viabilizar a um agente de *software* determinar automaticamente o seu significado e implementar tarefas como descoberta, requisição e composição de serviços automaticamente. Para solucionar este problema, a ferramenta SWS Editor descreve semanticamente Serviços Web *restful* que estejam descritos sintaticamente em WSDL com o objetivo de facilitar as tarefas de descoberta, requisição e composição automática de serviços. Além de implementar outros requisitos, e.g.: (i) utilização de ontologias e Serviços Web existentes na Internet; (ii) criação de repositórios de ontologias, Serviços Web semânticos no qual serão disponibilizados na web, (iii) criação de um recomendador de ontologias baseado no repositório local. Para realizar a anotação semântica dos Serviços Web *restful* foi utilizado a especificação SAWSDL em conjunto com ontologias construídas em linguagem OWL. A anotação semântica é facilitada pela ferramenta, pois fornece diversas funcionalidades para o usuário como, por exemplo, um editor para alteração dos documentos WSDL e *XML Schema Defintion* e suporte ao *context menu* para as funcionalidades de edição, inclusão e remoção da semântica em tais documentos.

Com o objetivo de atender a esses requisitos foi realizada uma avaliação, no Capítulo 2, sobre tecnologias e padrões da Web Semântica que são recomendações do W3C (e.g.: OWL, RDF), desenvolvimento de serviços e Serviços Web, Serviços Web Semânticos que subsidiaram a elaboração da ferramenta SWS Editor. Foram discutidos no Capítulo 3 propostas de trabalhos semelhantes em relação a anotação semântica de Serviços Web que possibilitaram estabelecer os requisitos apresentados. A partir dessas avaliações foi apresentada a arquitetura e serviços oferecidos pela ferramenta SWS Editor (Capítulo 4) que possibilitaram realizar a anotação semântica de acordo com os objetivos propostos no

Capítulo 1 e requisitos estabelecidos no Capítulo 3. Um exemplo de uso (Capítulo 5), realizado a partir de testes caixa preta, foi conduzido com o intuito de ilustrar o acesso aos serviços implementados e realizar a anotação semântica dos Serviços Web.

O uso da especificação SAWSDL favorece ao aumento de produtividade na realização da anotação semântica, pois através do documento WSDL é descrito tanto as funcionalidades sintáticas como semânticas dos Serviços Web. Por essa razão, os desenvolvedores de *software* não necessitam aprender uma nova linguagem para construir serviços semânticos facilitando o processo de anotação semântica.

6.2 PRINCIPAIS CONTRIBUIÇÕES

Com o desenvolvimento desta dissertação compreende-se que foram alcançados os seguintes benefícios:

a) Desenvolvimento baseado em Serviços

A arquitetura da ferramenta SWS Editor, baseada em serviços, permite a utilização da ferramenta SWS Editor em diferentes ambientes, e.g. aplicações *mobile*, *web* ou *desktop*, promovendo dessa forma o seu reuso, sendo que nesta dissertação optou-se por aplicações cliente *desktop*. Agregando tecnologias que são recomendações da W3C, a ferramenta SWS Editor oferece vantagens que atendem tanto ao consumidor dos serviços existentes, quanto aos desenvolvedores de novos serviços que podem ser incorporados a arquitetura proposta, pois favorece ao uso de padrões aceitos pela indústria de *software*. Portanto, com base nessas características da ferramenta, pode-se afirmar que o seu uso para anotação semântica de Serviços Web fornece os seguintes benefícios:

- Reusabilidade: Os serviços disponibilizados pela ferramenta SWS Editor podem ser utilizados em diferentes contextos de processos de negócios e interorganizacionais, por utilizar padrões abertos e recomendados pelo W3C como o XML, WSDL, *XML Schema Definition*, SAWSDL, e por essa razão, podem ser utilizados em diversos cenários de consumo;
- Baixo Acoplamento: Os serviços da ferramenta SWS Editor poderão evoluir ao longo do tempo sem causar impactos aos consumidores do serviço. Nesse caso, alterações realizadas nos serviços requisitados, por exemplo, por consumidores novos não deverão impactar em consumidores mais antigos. Por essa razão, a ferramenta SWS Editor realiza o versionamento das suas operações. Esse

benefício favorece a agilidade nos negócios, pois possibilita às organizações responderem rapidamente as mudanças no ambiente organizacional;

- Interoperabilidade: Por utilizar padrões abertos baseados em XML e recomendados pelo W3C, quaisquer que sejam a linguagem de programação, plataforma ou sistema operacional poderão ser utilizados para consumir os serviços da ferramenta SWS Editor.

b) Anotação semântica de documentos de documentos WSDL 2.0

- A anotação semântica realizada diretamente em documentos WSDL, de acordo com a especificação SAWSDL, possibilita a criação de Serviços Web Semânticos *restful* e também dos Serviços Web SOAP. Por ser uma recomendação do W3C, serviços que não possuam descrição sintática realizadas em documentos WSDL poderão realizar a conversão das suas interfaces com o objetivo de adquirir semântica através da ferramenta SWS Editor.

c) Acesso a ontologias disponíveis na web

- O acesso às ontologias disponíveis na web favorecem na obtenção de um número maior de documentos OWL para determinar o vocabulário semântico dos documentos WSDL.

d) Acesso à Serviços Web disponíveis na web

- Os acessos aos Serviços Web disponíveis na web favorecem na busca de documentos WSDL flexibilizando a obtenção desses serviços, pois os documentos não necessitam previamente existirem no servidor local.

e) Recomendador de Ontologias

- Este trabalho iniciou o desenvolvimento de um recomendador de ontologias para realizar anotação de semântica de Serviços Web similares aos já anotados pela ferramenta, entretanto, o recomendador proposto não implementa os algoritmos de recomendação que oferecem como vantagem a automação na tarefa da anotação semântica e será oferecido em um trabalho futuro.

f) Repositório Semântico

- Definição de um repositório semântico baseado nas anotações semânticas dos documentos WSDL que possibilitará a realização de buscas semânticas, integração de serviços.

Os resultados alcançados nesta dissertação foram baseados nos objetivos apresentados no Capítulo 1:

a) O estudo sobre a Web Semântica possibilitou a utilização de tecnologias como a especificação SAWSDL e utilização de ontologia OWL utilizada na ferramenta SWS Editor para o fornecimento da anotação semântica nos documentos WSDL 2.0. A utilização da linguagem SPARQL a partir do *framework Jena* (APENDICE F) possibilitou demonstrar a inclusão e recuperação no repositório semântico;
b) O estudo sobre Serviços e Serviços Web possibilitou projetar a arquitetura da ferramenta que possa ser utilizada por diferentes ambientes. Nesta dissertação, optou-se em acessar os serviços a partir de um ambiente *desktop* conforme exemplificado no Capítulo 5. Os documentos WSDL que representam cada serviço construído está disponível nos APENDICES G, H e I.
c) As pesquisas realizadas sobre Serviços Web semânticos *restful* possibilitaram identificar diferentes tecnologias de implementação para essas aplicações de *software*, e.g., SA-REST utilizado para fornecer semântica a Serviços Web *restful* que possuem descrição sintática em documentos HTML/XHTML, entretanto, neste trabalho optou por descrever semanticamente os Serviços Web *restful* que possuam sua descrição sintática disponibilizada em documentos WSDL 2.0 por ser uma recomendação do W3C;
d) A elaboração da ferramenta baseada em serviços possibilitou o acesso a Serviços Web e ontologias disponíveis na web. As ontologias OWL fornecem suporte a determinação do vocabulário semântico dos Serviços Web;
e) O exemplo de uso possibilitou a realização de testes caixa preta para demonstrar o acesso aos serviços oferecidos pela ferramenta.

Os artefatos gerados para elaboração desta dissertação estão disponíveis em: https://github.com/cleberlira/SWS-Editor.

6.3 PUBLICAÇÕES RELACIONADAS A ESTE TRABALHO

Durante o desenvolvimento deste trabalho, artigos foram submetidos para conferências especializadas em Sistemas de Informação, Internet e Futuro da Internet com o propósito de validar, através da comunidade acadêmica, as contribuições obtidas com o desenvolvimento desta dissertação. As publicações são listadas a seguir:

a) Lira de Santana, Cleber Jorge and Silva, Paulo Caetano da. (2015). A comparative study of tools that perform the semantic annotation of web services. 12th International Conference on Information Systems and Technology Management – CONTECSI – May, 20 to 22, 2015 – São Paulo, Brazil

b) Lira de Santana, Cleber Jorge and Silva, Paulo Caetano da. (2015). SWS Editor: A service-based tool for semantic annotation of RESTFul Web Services. 14th International Conference on WWW/INTERNET 24 – 26 October, 2015 - Maynooth, Greater Dublin, Ireland

6.4 TRABALHOS FUTUROS

Ao concluir este trabalho, percebeu-se algumas lacunas as quais necessitam serem estudadas com o objetivo de aperfeiçoar a construção de Serviços Web Semânticos e ampliar as pesquisas na área da Web Semântica. Os trabalhos futuros identificados nesta pesquisa estão separados em dois grupos: (i) trabalhos de implementação, relacionados ao acréscimo de novas funcionalidades na ferramenta SWS Editor e (ii) trabalhos de inovação, relacionados a continuidade da pesquisa desenvolvida nesta dissertação.

a) Trabalhos de implementação:

- anotação semântica em documentos WADL, é uma linguagem para descrição sintática de aplicações baseadas no protocolo HTTP, a exemplo dos Serviços Web *restful*. Por essa razão, a inclusão dessa funcionalidade na ferramenta SWS Editor, favoreceria a implementação de Serviços Web *restful* semânticos;
- anotação semântica em documentos HTML, de acordo com Maleshkova, Pedrinaci e Domingue (2009), muitos serviços da Web 2.0 descrevem suas informações sintáticas em documentos HTML, o que irá aumentar o número de aplicações suportada pela ferramenta SWS Editor;
- em uma nova versão da ferramenta o serviço *Ontology Recommender* deverá implementar algoritmos de recomendação com o objetivo de fornecer um grau

maior de automação na identificação das ontologias mais apropriadas na descrição dos Serviços Web, pois esses algoritmos atuarão fornecendo sugestões para o usuário do SWS Editor dos conceitos ontológicos que possam ser utilizados na anotação semântica;

- Possibilitar a utilização de ontologias web descritas em outras linguagens como a WSML. A linguagem WSML é uma linguagem usada para descrever modelos semânticos em um domínio específico;
- Descrever semanticamente APIs de Redes Sociais. Para implementar este requisito é necessário que essas APIs possuam sua descrição sintática em WSDL. Caso não haja essa descrição, uma biblioteca pode ser construída para o fornecimento da descrição sintática dessas APIs, o que possibilitaria realizar a anotação semântica dessas aplicações através da SWS Editor;
- É esperado realizar testes de performance (e.g.: carga, stress) com o objetivo de estimar o comportamento da ferramenta SWS Editor em um ambiente de produção através do qual o número de usuários é maior;
- Criar um serviço para extrair as anotações semânticas do documento WSDL, com o objetivo de favorecer o estabelecimento do repositório semântico para possibilitar posteriormente a descoberta e composição dos Serviços Web;
- Disponibilizar uma versão do SWS Editor em ambiente *mobile*;
- Disponibilizar a ferramenta para avaliar requisitos de interface homem máquina.

b) Trabalhos futuros relacionados a inovação

- Realizar um estudo para possibilitar que Serviços Web possam consultar dados *Linked Data*. Essa consulta pode ser realizada através de Serviços Web Semânticos. De acordo com (TRAN et al., 2014), *Linked Data* refere-se a Web de Dados, que possui os seus dados estruturados através de URIs possibilitando representar e acessar esses dados;
- Incluir o conceito de alinhamento de ontologias com o objetivo de abstrair a ontologia que está sendo utilizada.

REFERÊNCIAS

APACHE. **Apache Jena**. Disponível em: <https://jena.apache.org/>. Acesso em: 12 mar. 2015.

BAADER, Franz (Ed.). **The description logic handbook:** theory, implementation, and applications. Cambridge: Cambridge University Press, 2003.

BARROS, Aidil Jesus da Silveira; LEHFELD, Neide Aparecida de Souza. **Fundamentos de metodologia:** um guia para a iniciação científica. 3. ed. São Paulo: Pearson Prentice Hal, 2000.

BECKETT, D. et al. **RDF 1.1 Turtle–Terse RDF Triple Language. W3C Recommendation.** Disponível em: <http://www.w3.org/TR/turtle/ >**.** Acesso em: 12. fev; 2014.

BECKETT, Dave; MCBRIDE, Brian. RDF/XML syntax specification (revised). **W3C recommendation,** v. 10, 2004.

BERNERS-LEE, Tim; CONNOLLY, Dan. **Notation3 (N3): A readable RDF syntax. W3C Team Submission.** Disponível em: <http://www.w3.org/TeamSubmission>. Acesso em: 16 fev. 2014.

BIZER, Christian. The emerging web of linked data. Intelligent Systems, IEEE, v. 24, n. 5, p. 87-92, 2009.

BORGES, FERNANDO CEZAR REIS. **WS-PBRF: um framework de serviços para o processamento de regras financeiras de negócio expressas com XBRL formula.** 2012. Dissertação (Mestrado) – Sistemas e Computação. Universidade Salvador – Unifacs, 2012.

BRICKLEY, Dan; GUHA, R. V.; MCBRIDE, B. **RDF Schema 1.1, W3C Recommendation. World Wide Web Consortium**. Disponível em: <http://www.w3.org/TR/rdf-schema/>. Acesso em: 12 jna. 2014.

BUDINOSKI, Koste et al. An application for semantic annotation of Web services. In: INTERNATIONAL CONFERENCE FOR INFORMATICS AND INFORMATION TECHNOLOGY, 7., 2010. **Proccedings… 2**010.

CARDOSO, Jorge; PINTO, Alexandre Miguel. The Web Ontology Language (OWL) and its Applications. In: KHOSROW-POUR, Mehdi (Ed.). **Encyclopedia of Information Science and Technology, Information Science Pub**, 2015. Disponível em: < http://jorge-cardoso.github.io/publications/Papers/BC-2015-031-ISR-OWL-and-Its-Applications.pdf>. Acesso em: 16 fev. 2014.

CHAPPELL, David A.; JEWELL, Tyler. **Java web services**. [S.l.]: Tecniche Nuove, 2002.

CHOI, Jae; NAZARETH, Derek L.; JAIN, Hemant K.. Implementing Service-Oriented Architecture in Organizations. **Journal of Systems and Software**, New York, v.26, p. 253-286, 2010. Disponível em: <http://dl.acm.org/citation.cfm?id=2069458>.: Acesso em: 12 abr. 2014.

CLARK, K. G.; FEIGENBAUM, L.; TORRES, E. **SPARQL protocol for RDF. W3C Recommendation.** 2008. Disponível em: <http://www.w3.org/TR/rdf-sparql-protocol/>. Acesso em: 12 nov. 2014.

DAMASCENO, Julio Cesar. Introdução à Composição de Serviços Web. 7 In: ERCEMAPI 2009, Teresina. **Anais...** Teresina: SBC, 2009. Disponível em:<http://www.ufpi.br/subsiteFiles/ercemapi/arquivos/files/minicurso/mc8.pdf>. Acesso

em: 12 mar. 2014.

DIMITROV, Marin et al. WSMO studio–A semantic web services modelling environment for WSMO. In: EUROPEAN SEMANTIC WEB CONFERENCE, 2007, Berlin**. Proccedings**… 2007. p. 749-758.

ERL, Thomas. **Service-oriented architecture concepts, technology and design**. 11.ed. Boston, United States of America : Prentice Hall. 2011. 759p.

FERREIRA FILHO, Otávio Freitas. **Serviços semânticos:** uma abordagem RESTful. 2011. Tese (Doutorado)- Universidade de São Paulo USP, São Paulo, 2011.

FIELDING, Roy Thomas. **Architectural styles and the design of network-based software architectures**. 2000. Tese de Doutorado. University of California, Irvine.

FUGITA, HENRIQUE SHOITI; HIRAMA, Kwchi. **SOA:** modelagem, análise e design. Rio de Janeiro: Elsevier 2012.

GEARON, P.; PASSANT, A.; POLLERES, A. SPARQL 1.1 Update. W3C Recommendation, 21 March 2013. In: WORLD WIDE WEB CONSORTIUM, 2013. **Proccedings**… 2013. <Disponível em: http://www.w3.org/TR/sparql11-update, 2013.>

GUTTULA, Chaitanya. RADIANTWEB: **A tool facilitating semantic annotation of web services.** 2012. Tese (Doutorado)- University of Georgia, 2012.

HADLEY, Marc J. **Web application description language (WADL).** [S.l.]: [s.n.], 2006.

HARRIS, Steve; SEABORNE, Andy. **SPARQL 1.1 query language. W3C Recommendation**, v. 21, 2013.

LARMAN, Craig. **Utilizando UML e Padrões**: uma introdução à análise e ao projeto orientado a objeto e ao desenvolvimento interativo. 3.ed. Porto Alegre, RS : Artmed : Bookman. 2007.

KALIN, Martin. **Java Web Services**: Rio de Janeiro, RJ: Alta Books. 2010.

KAY, Michael et al. Xsl transformations (xslt) version 2.0. **W3c recommendation**, v. 23, p. 52-71, 2007.

KLUSCH, Matthias; KAPAHNKE, Patrick. Semantic web service selection with SAWSDL-MX**.** In: INTERNATIONAL SEMANTIC WEB CONFERENCE, 7., 2008. **Proccedings…** 2008.p. 3.

KLUSCH, Matthias; KAPAHNKE, Patrick; ZINNIKUS, Ingo**.** Adaptive hybrid semantic selection of SAWSDL services with SAWSDL-MX2. In: SEMANTIC-ENABLED Advancements on the Web: Applications Across Industries: Applications Across Industries [S.l.]: [s.n.], 2012.

KUMAR, B.V;NARAYAN, P.; NG,T. **Implementing SOA Using Java J2EE**. Boston: [s.n.], 2006.

KLYNE, Graham; CARROLL, Jeremy J. **Resource description framework (RDF):** Concepts and abstract syntax. [S.l.]: [s.n.], 2006.

KOPECKY, Jacek et al. Sawsdl: Semantic annotations for wsdl and xml schema**. Internet Computing, IEEE**, v. 11, n. 6, p. 60-67, 2007.

KOPECKY, Jacek; GOMADAM, Karthik; VITVAR, Tomas. Hrests: An html microformat for describing restful web services. In: WEB INTELLIGENCE AND INTELLIGENT

AGENT TECHNOLOGY, 2008. WI-IAT'08. IEEE/WIC/ACM INTERNATIONAL

CONFERENCE ON. IEEE, 2008. **Proccedings**… 2008. p. 619-625.

LEENHEER, P. D. **Open standards for the semantic web: Xml/rdf (s)/owl/soap**. Brussel: Vrije Universiteit Brussel, 2009.

MALESHKOVA, Maria; KOPECKÝ, Jacek; PEDRINACI, Carlos. Adapting SAWSDL for semantic annotations of restful services. In: ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS: OTM 2009 WORKSHOPS, 2009, Berlin Heidelberg. **Proccedings**… 2009. p. 917-926.

MANDEL, Lawrence. **Describe REST Web services with WSDL 2.0** - Disponível em <http://www.ibm.com/developerworks/library/ws-restwsdl/>. Acesso em: 1 abr. 2014.

MARCONI, M.; LAKATOS, E. **Metodologia cientifica:** ciência e conhecimento científico. 5. ed. São Paulo: Atlas, 2001.

N.BOISSEL-DALLIER, J. Lorré ; BENABEN, F. Management Tool for Semantic Annotations in WSDL**.** In: ON THE MOVE TO MEANINGFUL INTERNET SYSTEMS: OTM 2009 WORKSHOPS, 2009. **Proccedings**… 2009.

O'REILLY, Tim. **What is Web 2.0.** 2005. Disponível em <http://s3.amazonaws.com/academia.edu.documents/38552727/OReilly_Radar_-_Web_2.0_Compact_Definition.pdf?AWSAccessKeyId=AKIAJ56TQJRTWSMTNPEA&Expires=1466347028&Signature=FFtNoRp1fGJl5Z8a8AZAIMp2Ha4%3D&response-content-disposition=inline%3B%20filename%3DWeb_2.0_Compact_Definition.pdfAcesso em: 1 abr. 2014.

DE OLIVEIRA, Douglas Manoel; MENEGAZZO, Cinara Terezinha; CLARO, Daniela Barreiro**. Uma análise conceitual das linguagens semânticas de serviços web focando nas composições:** comparação entre OWL-S, WSMO e SAWSDL. 2009. [S.l.]: [s.n.], 2006.

PANZIERA, L. et al WSML or OWL? A lesson learned by addressing NFP-based selection of semantic Web services. In: NON FUNCTIONAL PROPERTIES AND SLA MANAGEMENT (NFPSLAM 2010) WORKSHOP. 2010. **Proccedings**… 2010.

PATIL, Abhijit A. et al. Meteor-s web service annotation framework. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB. ACM, 13., 2004. **Proccedings**… 2004. p. 553-562.

PAUTASSO, Cesare; ZIMMERMANN, Olaf; LEYMANN, Frank. Restful web services vs. big'web services: making the right architectural decision. In: INTERNATIONAL CONFERENCE ON WORLD WIDE WEB. ACM, 17., 2008. **Proccedings**… 2008. p. 805-814.

PRAZERES, Cássio Vinícius Serafim. **Serviços Web Semânticos:** da modelagem à composição. 2009. Tese (Doutorado)- Universidade de São Paulo – USP, São Paulo, 2009..

PRESSMAN, R. S. **Engenharia de Software**. 3.ed. São Paulo, SP, Brasil : Pearson Makron Books. 2007.

RICHARDSON, Leonard; RUBY, Sam. **RESTful web services**. [S.l.]: O'Reilly Media, Inc., 2008.

SEGEV, Aviv; SHENG, Quan Z. Ontology construction for web services. In: WORKSHOP ON ONTOLOGY PATTERNS (WOP 2009), HELD TOGETHER WITH INTERNATIONAL SEMANTIC WEB CONFERENCE (ISWC 2009), 8., 2009. **Proccedings**… 2009. p. 147-154.

SNELL, James. **Resource-oriented vs. activity-oriented Web services. 2004**. Disponível em: <http://www.ibm.com/developerworks/webservices/library/ws-restvsoap/>. Acesso em: 15 set. 2013.

SOMMERVILLE, Ian. **Engenharia de Software** 8. ed. São Paulo, SP : PearsonAddison-Wesley. 2007.

STAVROPOULOS, Thanos G.; VRAKAS, Dimitris; VLAHAVAS, Ioannis. **Iridescent:** a Tool for Rapid Semantic Web Service Descriptions. [S.l.]: [s.n.], 2011.

TORRES, Carlos Eduardo Atencio. **Uso de informação linguística e análise de conceitos formais no aprendizado de ontologias.** 2012. Tese (Doutorado)- Universidade de São Paulo-USP, 2102.

TRAN, Tuan Nhat et al. Linked Data Mashups: A Review on Technologies, Applications and Challenges. In: INTELLIGENT Information and Database Systems. Springer International Publishing, 2014. p. 253-262.

TYAGI, Sameer. **RESTful Web Services**. <Disponível em: http://www.oracle.com/technetwork/articles/javase/index-137171.html>. Acesso em: 4 dez. 2014.

VIANA, Phillip Luiz. **Uso de composição automática de serviços e similaridade para análise de integração de processos de negócio.** 2013. Tese (Doutorado)-Universidade de São Paulo – USP, 2013.

W3C. **OWL Web Ontology Language**. Disponível em: <http://www.w3.org/TR/owl-features>. Acesso em: 10 ago. 2013.

W3C. **RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation 2014.** Disponível em <http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>. Acesso em: 27 set. 2014.

W3C. **XML**. <Disponível em: http://www.w3.org/XML/>. Acesso em: 15 jan. 2014.

WSDL 2.0 - Overview. Disponível em http://tutorials.jenkov.com/wsdl/overview.html. Acesso em: 10 ago. 2014.

XAVIER, Otávio Calaça. **Serviços Web Semânticos Baseados em RESTful**. 2011. Tese (Doutorado)- Universidade Federal de Goiás, 2011.

APENDICE A - TRECHO DE UM *WEB SERVICE RESTFUL* ANOTADO COM SAWSDL UTILIZANDO A FERRAMENTA SWS EDITOR

1.<?xml version="1.0" encoding="UTF-8"?>

2.<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"

3. xmlns:sawsdl="http://www.w3.org/ns/sawsdl">

4. ...

5. <wsdl:interface name="FlightListInterface" sawsdl:**modelReference**="http://example.data/flight.owl# #FlightName ">

6. <wsdl:operation name="getFlightList"

7.sawsdl:**modelReference**=" http://example.com/data/flight.owl# flightList#RequestFlightList"

8. pattern="http://www.w3.org/ns/wsdl/in-out"

9. style="http://www.w3.org/ns/wsdl/style/iri" wsdlx:safe="true">

10. <wsdl:input element="msg:getFlightList"/>

11. <wsdl:output element="msg:flightList"/>

12. </wsdl:operation>

13. </wsdl:interface>

14. <wsdl:binding name="FlightListHTTPBinding"

15. type=http://www.w3.org/ns/wsdl/http interface="tns:FlightListInterface">

16..<wsdl:operation ref="tns:getFlightList" whttp:method="GET"

17.sawsdl:**modelReference**="http://example.com/data/flight.owl#RequestGetFlightList"/>

18. </wsdl:binding>

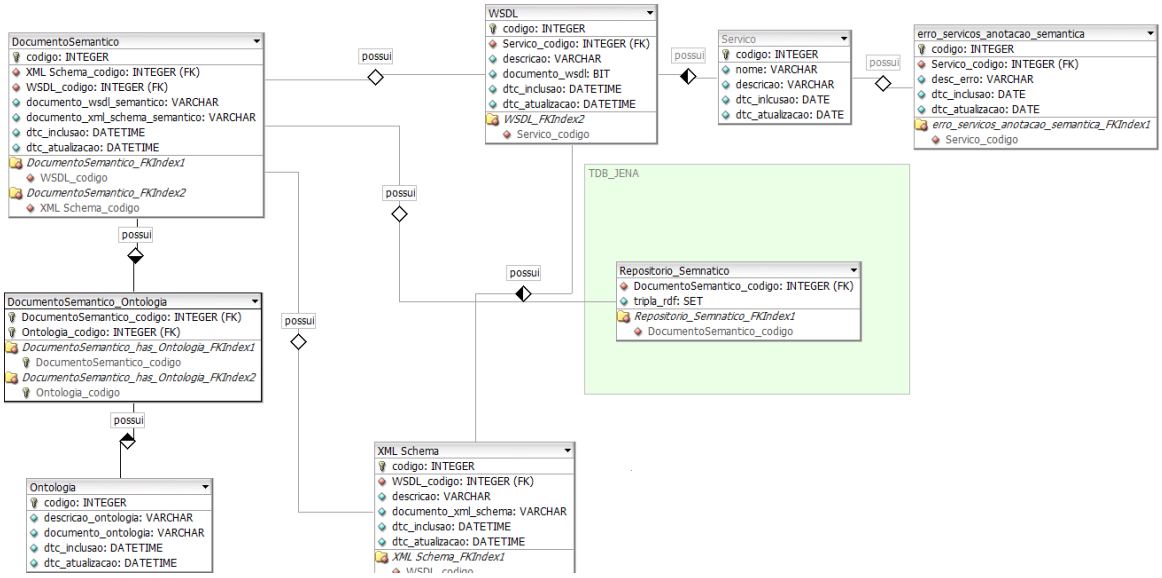
19. ...

20. </wsdl:description>

A consulta abaixo recupera os serviços de voo que possuem o nome "FlightName".

1.PREFIX SAWSDL:<HTTP://EXAMPLE.DATA/FLIGHT.OWL>
2.SELECT DISTINCT ?FLIGHTS
3.WHERE {
4.?FLIGHTS SAWSDL:MODELREFERENCE
5.<HTTP://EXAMPLE.DATA/FLIGHT.OWL#FlightName >
6.}

APENDICE B - MODELAGEM E SCRIPT DE DADOS DA FERRAMENTA SWS EDITOR



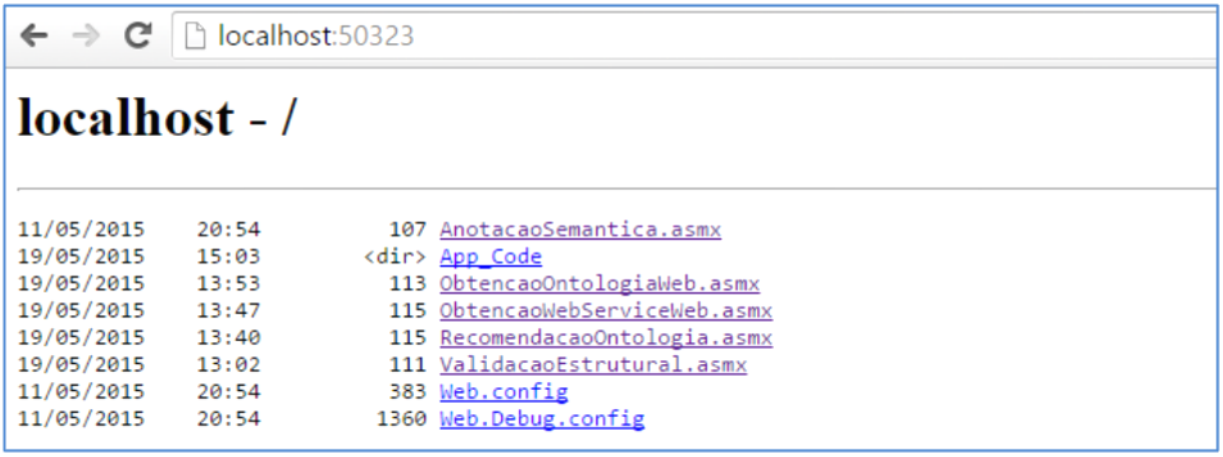

```
CREATE TABLE DocumentoSemantico (
  codigo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  XML Schema_codigo INTEGER UNSIGNED NOT NULL,
  WSDL_codigo INTEGER UNSIGNED NOT NULL,
  documento_wsdl_semantico VARCHAR NULL,
  documento_xml_schema_semantico VARCHAR NULL,
  dtc_inclusao DATETIME NULL,
  dtc_atualizacao DATETIME NULL,
  PRIMARY KEY(codigo),
  INDEX DocumentoSemantico_FKIndex1(WSDL_codigo),
  INDEX DocumentoSemantico_FKIndex2(XML Schema_codigo)
);

CREATE TABLE DocumentoSemantico_Ontologia (
  DocumentoSemantico_codigo INTEGER UNSIGNED NOT NULL,
```

```
  Ontologia_codigo INTEGER UNSIGNED NOT NULL,
  PRIMARY KEY(DocumentoSemantico_codigo, Ontologia_codigo),
  INDEX DocumentoSemantico_has_Ontologia_FKIndex1(DocumentoSemantico_codigo),
  INDEX DocumentoSemantico_has_Ontologia_FKIndex2(Ontologia_codigo)
);

CREATE TABLE erro_servicos_anotacao_semantica (
  codigo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Servico_codigo INTEGER UNSIGNED NOT NULL,
  desc_erro VARCHAR NULL,
  dtc_inclusao DATE NULL,
  dtc_atualizacao DATE NULL,
  PRIMARY KEY(codigo),
  INDEX erro_servicos_anotacao_semantica_FKIndex1(Servico_codigo)
);

CREATE TABLE Ontologia (
  codigo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  descricao_ontologia VARCHAR NULL,
  documento_ontologia VARCHAR BINARY NULL,
  dtc_inclusao DATETIME NULL,
  dtc_atualizacao DATETIME NULL,
  PRIMARY KEY(codigo)
);

CREATE TABLE Repositorio_Semnatico (
```

```
  DocumentoSemantico_codigo INTEGER UNSIGNED NOT NULL,
  tripla_rdf SET NULL,
  INDEX Repositorio_Semnatico_FKIndex1(DocumentoSemantico_codigo)
);

CREATE TABLE Servico (
  codigo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  nome VARCHAR NULL,
  descricao VARCHAR NULL,
  dtc_inlcusao DATE NULL,
  dtc_atualizacao DATE NULL,
  PRIMARY KEY(codigo)
);

CREATE TABLE WSDL (
  codigo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  Servico_codigo INTEGER UNSIGNED NOT NULL,
  descricao VARCHAR NULL,
  documento_wsdl BIT NULL,
  dtc_inclusao DATETIME NULL,
  dtc_atualizacao DATETIME NULL,
  PRIMARY KEY(codigo),
  INDEX WSDL_FKIndex2(Servico_codigo)
);
```

```
CREATE TABLE XML_Schema (
  codigo INTEGER UNSIGNED NOT NULL AUTO_INCREMENT,
  WSDL_codigo INTEGER UNSIGNED NOT NULL,
  descricao VARCHAR NULL,
  documento_xml_schema VARCHAR BINARY NULL,
  dtc_inclusao DATETIME NULL,
  dtc_atualizacao DATETIME NULL,
  PRIMARY KEY(codigo),
  INDEX XML Schema_FKIndex1(WSDL_codigo)
);
```

APENDICE C - LISTA DE SERVIÇOS NO SERVIDOR WEB



APENDICE D - JAVADOC PARA CRIAÇÃO, INCLUSÃO E CONSULTAS NO REPOSITÓRIO SEMÂNTICO

Class LivrosSAWSDL

java.lang.Object

LivrosSAWSDL

public class **LivrosSAWSDL**
extends java.lang.Object

Version:

1.0

Author:

Cleber Lira

See Also:

março/2015

Constructor Summary

Constructors

Constructor and Description
LivrosSAWSDL()

- ***Method Summary***

All Methods Static Methods Instance Methods Concrete Methods

Modifier and Type	Method and Description
Void	Executarconsulta(java.lang.String query)
Void	incluir(java.lang.String triplaRDF)
static void	main(java.lang.String[] args)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

- *LivrosSAWSDL*

public LivrosSAWSDL()

- ***Method Detail***

- *incluir*

public void incluir(java.lang.String triplaRDF)

Parameters:

triplaRDF -

- *Executarconsulta*

public void Executarconsulta(java.lang.String query)

Parameters:

triplaRDF -

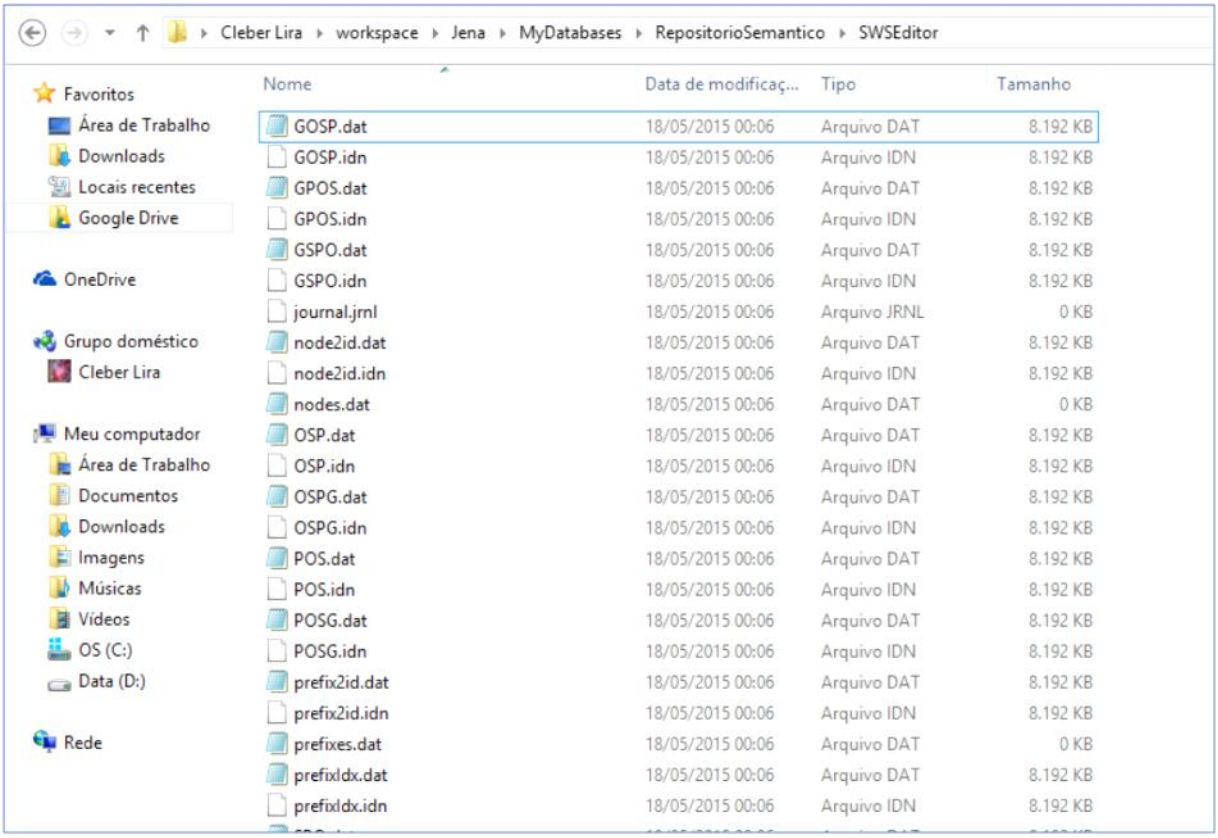
APENDICE E - CÓDIGO FONTE PARA DEFINIÇÃO DO REPOSITÓRIO SEMÂNTICO

```
import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.tdb.TDBFactory;
import com.hp.hpl.jena.update.GraphStore;
import com.hp.hpl.jena.update.GraphStoreFactory;
import com.hp.hpl.jena.update.UpdateAction;
import java.util.logging.Logger;

/***
 * @author Cleber Lira
 * @version 1.0
 * @see março/2015
 */

public class Repositorio {

	private static final Logger LOGGER = Logger.getLogger(Repositorio.class.getName());
	/**
	 * @author Cleber
	 * @param triplaRDF
	 * @return void
	 */
	public void incluir(String triplaRDF  ){
		String directory = "MyDatabases/RepositorioSemantico/SWSEditor" ;
		Dataset dataset = TDBFactory.createDataset(directory) ;
		GraphStore graphStore=GraphStoreFactory.create(dataset);
		UpdateAction.parseExecute(triplaRDF, graphStore);
	}

	/**
	 * @author Cleber
```

```
     * @param triplaRDF
          * @return void
          */
         public void Executarconsulta(String queryString){
                  String directory = "MyDatabases/RepositorioSemantico/SWSEditor" ;
                  Dataset dataset = TDBFactory.createDataset(directory) ;
                  GraphStore graphStore=GraphStoreFactory.create(dataset);
                  Dataset d = graphStore.toDataset();
            //Busca semÂntica
                  LOGGER.info("_____________________________________________________");
            Query query2 = QueryFactory.create(queryString);
            QueryExecution qe = QueryExecutionFactory.create(query2, d);
                  ResultSet results = qe.execSelect();

                  ResultSetFormatter.out(System.out, results, query2);

                  qe.close();
         }

}

import com.hp.hpl.jena.query.Dataset;
import com.hp.hpl.jena.query.Query;
import com.hp.hpl.jena.query.QueryExecution;
import com.hp.hpl.jena.query.QueryExecutionFactory;
import com.hp.hpl.jena.query.QueryFactory;
import com.hp.hpl.jena.query.ResultSet;
import com.hp.hpl.jena.query.ResultSetFormatter;
import com.hp.hpl.jena.tdb.TDBFactory;
import com.hp.hpl.jena.update.GraphStore;
import com.hp.hpl.jena.update.GraphStoreFactory;
import com.hp.hpl.jena.update.UpdateAction;

import java.util.logging.Logger;

public class AppRepositorio {
         private static final Logger LOGGER = Logger.getLogger(AppRepositorio.class.getName());

         public static void main(String[] args) {
```

```
                Repositorio repositorio = new Repositorio();
                String triplaRDF="PREFIX sawsdl: <http://example.edu.unifacs/ontologies/bookstore.owl#Technologies> INSERT DATA{<http://example/categorization/books> sawsdl: 'Java'}";
                LOGGER.info("Inclusão no repositório Semântico");
                repositorio.incluir(triplaRDF);

                triplaRDF = "PREFIX sawsdl: <http://example.edu.unifacs/ontologies/bookstore.owl#Technologies> INSERT DATA{<http://example/categorization/books> sawsdl: 'C#'}";

                repositorio.incluir(triplaRDF);

                String queryString =

          "PREFIX sawsdl: <http://example.edu.unifacs/ontologies/bookstore.owl#Technologies>" +
          "SELECT ?livro " +
          " WHERE {" +
          " <http://example.edu.unifacs/ontologies/bookstore.owl#Technologies> <http://example/categorization/books> ?livro .}";
        LOGGER.info("Realizando consulta repositório Semântico");
        repositorio.Executarconsulta(queryString);
      }
}
```

APENDICE F - REPOSITÓRIO SEMÂNTICO



APENDICE G - TRECHO DO CONTRATO WSDL DOS SERVIÇOS STRUCTURAL VALIDATION E OBTAINING ONTOLOGIES ON THE WEB

```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/" xmlns:tns="http://unifacs.edu/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:s="http://www.w3.org/2001/XMLSchema"
xmlns:soap12="http://schemas.xmlsoap.org/wsdl/soap12/" xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" targetNamespace="http://unifacs.edu/">
<wsdl:types>...</wsdl:types>
<wsdl:message name="realizarValidacaoEstruturalSoapIn">
<wsdl:part name="parameters" element="tns:realizarValidacaoEstrutural"/>
</wsdl:message>
<wsdl:message name="realizarValidacaoEstruturalSoapOut">
<wsdl:part name="parameters" element="tns:realizarValidacaoEstruturalResponse"/>
</wsdl:message>
<wsdl:portType name="ValidacaoEstruturalSoap">...</wsdl:portType>
<wsdl:binding name="ValidacaoEstruturalSoap" type="tns:ValidacaoEstruturalSoap">
<wsdl:binding name="ValidacaoEstruturalSoap12" type="tns:ValidacaoEstruturalSoap">
<soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="realizarValidacaoEstrutural">
<soap12:operation soapAction="http://unifacs.edu/realizarValidacaoEstrutural" style="document"/>
<wsdl:input>
<wsdl:output>
<soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ValidacaoEstrutural">...</wsdl:service>
</wsdl:definitions>
```

```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
<wsdl:types>...</wsdl:types>
<wsdl:message name="obterOntologiaWebSoapIn">
<wsdl:part name="parameters" element="tns:obterOntologiaWeb"/>
</wsdl:message>
<wsdl:message name="obterOntologiaWebSoapOut">
<wsdl:part name="parameters" element="tns:obterOntologiaWebResponse"/>
</wsdl:message>
<wsdl:portType name="ObtencaoOntologiaWebSoap">
<wsdl:operation name="obterOntologiaWeb">
<wsdl:input message="tns:obterOntologiaWebSoapIn"/>
<wsdl:output message="tns:obterOntologiaWebSoapOut"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="ObtencaoOntologiaWebSoap" type="tns:ObtencaoOntologiaWebSoap">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="obterOntologiaWeb">
<soap:operation soapAction="http://unifacs.edu/obterOntologiaWeb" style="document"/>
<wsdl:input>
<wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:binding name="ObtencaoOntologiaWebSoap12" type="tns:ObtencaoOntologiaWebSoap">...</wsdl:binding>
<wsdl:service name="ObtencaoOntologiaWeb">
<wsdl:port name="ObtencaoOntologiaWebSoap" binding="tns:ObtencaoOntologiaWebSoap">
<wsdl:port name="ObtencaoOntologiaWebSoap12" binding="tns:ObtencaoOntologiaWebSoap12">
<soap12:address location="http://localhost:50323/ObtencaoOntologiaWeb.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

APENDICE H - TRECHO DO CONTRATO WSDL DOS SERVIÇOS ONTOLOGY RECOMMENDER E OBTAINING WEB SERVICES ON THE WEB

```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 <wsdl:types>...</wsdl:types>
<wsdl:message name="realizarRecomendacaoSoapIn">
 <wsdl:part name="parameters" element="tns:realizarRecomendacao"/>
</wsdl:message>
<wsdl:message name="realizarRecomendacaoSoapOut">
 <wsdl:part name="parameters" element="tns:realizarRecomendacaoResponse"/>
</wsdl:message>
 <wsdl:portType name="RecomendacaoOntologiaSoap">...</wsdl:portType>
 <wsdl:binding name="RecomendacaoOntologiaSoap" type="tns:RecomendacaoOntologiaSoap">...</wsdl:binding>
<wsdl:binding name="RecomendacaoOntologiaSoap12" type="tns:RecomendacaoOntologiaSoap">
 <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="realizarRecomendacao">
 <soap12:operation soapAction="http://unifacs.edu/realizarRecomendacao" style="document"/>
<wsdl:input>
 <soap12:body use="literal"/>
</wsdl:input>
<wsdl:output>
 <soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="RecomendacaoOntologia">
<wsdl:port name="RecomendacaoOntologiaSoap" binding="tns:RecomendacaoOntologiaSoap">
 <soap:address location="http://localhost:50323/RecomendacaoOntologia.asmx"/>
</wsdl:port>
<wsdl:port name="RecomendacaoOntologiaSoap12" binding="tns:RecomendacaoOntologiaSoap12">
 <soap12:address location="http://localhost:50323/RecomendacaoOntologia.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
 <wsdl:types>...</wsdl:types>
<wsdl:message name="obterWebServiceWebSoapIn">
 <wsdl:part name="parameters" element="tns:obterWebServiceWeb"/>
</wsdl:message>
<wsdl:message name="obterWebServiceWebSoapOut">
 <wsdl:part name="parameters" element="tns:obterWebServiceWebResponse"/>
</wsdl:message>
 <wsdl:portType name="ObtencaoWebServiceWebSoap">...</wsdl:portType>
 <wsdl:binding name="ObtencaoWebServiceWebSoap" type="tns:ObtencaoWebServiceWebSoap">...</wsdl:binding>
<wsdl:binding name="ObtencaoWebServiceWebSoap12" type="tns:ObtencaoWebServiceWebSoap">
 <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="obterWebServiceWeb">
 <soap12:operation soapAction="http://unifacs.edu/obterWebServiceWeb" style="document"/>
<wsdl:input>
<wsdl:output>
 <soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ObtencaoWebServiceWeb">
<wsdl:port name="ObtencaoWebServiceWebSoap" binding="tns:ObtencaoWebServiceWebSoap">
 <soap:address location="http://localhost:50323/ObtencaoWebServiceWeb.asmx"/>
</wsdl:port>
<wsdl:port name="ObtencaoWebServiceWebSoap12" binding="tns:ObtencaoWebServiceWebSoap12">
 <soap12:address location="http://localhost:50323/ObtencaoWebServiceWeb.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

APENDICE I - TRECHO DO CONTRATO WSDL DOS SERVIÇOS ONTOLOGY RECOMMENDER E OBTAINING WEB SERVICES ON THE WEB

```
<wsdl:definitions xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/" xmlns:soapenc="http://schema
<wsdl:types>
<wsdl:message name="realizarAnotacaoSemanticaSoapIn">
<wsdl:message name="realizarAnotacaoSemanticaSoapOut">
<wsdl:portType name="AnotacaoSemanticaSoap">
<wsdl:binding name="AnotacaoSemanticaSoap" type="tns:AnotacaoSemanticaSoap">
<wsdl:binding name="AnotacaoSemanticaSoap12" type="tns:AnotacaoSemanticaSoap">
 <soap12:binding transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="realizarAnotacaoSemantica">
 <soap12:operation soapAction="http://unifacs.org/realizarAnotacaoSemantica" style="document"/>
<wsdl:input>
<wsdl:output>
 <soap12:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="AnotacaoSemantica">
<wsdl:port name="AnotacaoSemanticaSoap" binding="tns:AnotacaoSemanticaSoap">
 <soap:address location="http://localhost:50323/AnotacaoSemantica.asmx"/>
</wsdl:port>
<wsdl:port name="AnotacaoSemanticaSoap12" binding="tns:AnotacaoSemanticaSoap12">
 <soap12:address location="http://localhost:50323/AnotacaoSemantica.asmx"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

APENDICE J - IMPLEMENTAÇÃO DAS CLASSES

```
using System.Web.Services.Description;
using System.Windows.Forms;
using System.Xml;
using System.Xml.Serialization;
using Binding = System.Web.Services.Description.Binding;
using Message = System.Web.Services.Description.Message;

namespace ValidacaoEstrutural
{

public class Valida
{
public TreeNode ServiceNode;
private XmlSchemas _schemas;
private ServiceDescriptionCollection _services=new ServiceDescriptionCollection() ;

public Valida(ServiceDescription service)
{
_services.Add(service);
_schemas=service.Types.Schemas ;

if (service.Name == string.Empty)
        service.Name=service.RetrievalUrl;
  if (service.Name == string.Empty) service.Name=service.TargetNamespace;

        ServiceNode=new TreeNode(service.Name);
        ServiceNode.Tag=service.Documentation ;
        ServiceNode.ImageIndex=15;
        ServiceNode.SelectedImageIndex=15;
        Parse();
        ServiceNode.Expand ();
}

private string GetProtocol (Binding binding)
{
if (binding.Extensions.Find (typeof(SoapBinding)) != null) return "Soap";
HttpBinding hb = (HttpBinding) binding.Extensions.Find (typeof(HttpBinding));
        if (hb == null) return "";
                if (hb.Verb == "POST") return "HttpPost";
```

```
                if (hb.Verb == "GET") return "HttpGet";
                        return "";
}
```

```
private void GetOperationFormat(OperationBinding obin, Operation oper, out SoapBindingStyle style,out SoapBindingUse inputUse, out SoapBindingUse outputUse, out string requestMessage, out string responseMessage, out TreeNode requestNode,out TreeNode responseNode)
{
        style=SoapBindingStyle.Document;
        outputUse=SoapBindingUse.Literal;
        inputUse=SoapBindingUse.Literal;
        requestMessage=string.Empty;
        responseMessage=string.Empty;
        requestNode=null;
        responseNode=null;
        SoapBindingStyle pStyle;
        SoapBindingUse pInputUse, pOutputUse;

        GetOperationFormat (obin, out pStyle, out pInputUse, out pOutputUse  );
        if (oper.Messages.Input != null){
                requestNode=MessageToTreeNode(oper.Messages.Input,  pInputUse);
        }
        if (oper.Messages.Output != null){
        responseNode=MessageToTreeNode(oper.Messages.Output, pOutputUse  );
        }
                style=pStyle;
                outputUse=pOutputUse;
                inputUse=pInputUse;
        }
```

```
void GetOperationFormat (OperationBinding obin, out SoapBindingStyle style, out SoapBindingUse inputUse, out SoapBindingUse outputUse)
{
        style=SoapBindingStyle.Document;
        inputUse=SoapBindingUse.Literal;
        outputUse=SoapBindingUse.Literal;
        if (obin.Extensions != null) {
        SoapOperationBinding sob=obin.Extensions.Find (typeof(SoapOperationBinding)) as SoapOperationBinding;
        if (sob != null) {
        style = sob.Style;
```

```
        if (obin.Input != null) {
                SoapBodyBinding sbb0 = obin.Input.Extensions.Find (typeof(SoapBodyBinding)) as SoapBodyBinding;
        if (sbb0 != null)
          inputUse = sbb0.Use;
        }

if (obin.Output != null) {
        SoapBodyBinding sbb1 = obin.Output.Extensions.Find (typeof(SoapBodyBinding)) as SoapBodyBinding;
        if (sbb1 != null)
                outputUse = sbb1.Use;
        }
}
}}
private TreeNode MessageToTreeNode(OperationMessage omsg, SoapBindingUse use)
{
        Message msg=_services.GetMessage (omsg.Message);

        TreeNode node=new TreeNode() ;
        SchemaParser ngen=new SchemaParser(_schemas);
        ngen.BindingUse=use;

        foreach (MessagePart part in msg.Parts)
        {
                if (part.Element == XmlQualifiedName.Empty)
                  {
                        TreeNode partNode=ngen.Translate(part.Type);
                        partNode.ImageIndex=5;
                        partNode.SelectedImageIndex=5;
                        partNode.Text=part.Name;
                        node.Nodes.Add(partNode);
                   }
                else
                {
                TreeNode partNode=ngen.Translate(part.Element);
                partNode.ImageIndex=5;
                 partNode.SelectedImageIndex=5;

                 partNode.Text=part.Name;
                 node.Nodes.Add(partNode);
                 }
```

```
                }

                return node;
                }

public TreeNode TranslateOperation(Port port, OperationBinding obin, Operation oper, string protocol)
{
        TreeNode tnOperation=new TreeNode (oper.Name, 13 , 13);
        SoapBindingStyle style=new SoapBindingStyle() ;
        SoapBindingUse inputUse=new SoapBindingUse() ;
        SoapBindingUse outputUse=new SoapBindingUse() ;

        string requestmsg=string.Empty;
        string responsemsg=string.Empty;
        TreeNode tnInput=new TreeNode ();
        TreeNode tnOutput=new TreeNode ();
        TreeNode tnFault=new TreeNode ("Faults");

        GetOperationFormat (obin, oper, out style,out inputUse,out outputUse, out requestmsg, out responsemsg,out tnInput, out tnOutput);

        string operDesc=string.Empty;
        operDesc +=oper.Documentation + "\n";
        operDesc +="Style: " + style.ToString() + "\n" ;
        tnOperation.Tag=operDesc;
        MessageCollection messages=_services[0].Messages;
        if (oper.Messages.Input != null)
        {
        Message messageIn=messages[oper.Messages.Input.Message.Name] ;                    if (messageIn != null )
        {
        if (tnInput == null) tnInput=new TreeNode() ;
        tnInput.Tag=requestmsg;
        tnInput.ImageIndex=13;
        tnInput.SelectedImageIndex=13;
        if (oper.Messages.Input.Name != null && oper.Messages.Input.Name != string.Empty)
        {
                tnInput.Text="Input: " + oper.Messages.Input.Name ;
        }
        Else{
        tnInput.Text="Input: " + oper.Messages.Input.Message.Name ;
        if (tnInput != null) tnOperation.Nodes.Add  (tnInput );
```

```
};
};

if (oper.Messages.Output != null)
{
Message messageOut=messages[oper.Messages.Output.Message.Name] ;
if (messageOut != null )
{

if (tnOutput == null) tnOutput=new TreeNode() ;
tnOutput.Tag=responsemsg;
tnOutput.ImageIndex=13;
tnOutput.SelectedImageIndex=13;

if (oper.Messages.Output.Name != null &&     oper.Messages.Output.Name != string.Empty)
{
tnOutput.Text="Output: " + oper.Messages.Output.Name ;
}
else
tnOutput.Text="Output: " + oper.Messages.Output.Message.Name ;
if (tnOutput != null) tnOperation.Nodes.Add  (tnOutput );
};

};

foreach (OperationFault faultOp in  oper.Faults)
{
Message messageFault=messages[faultOp.Message.Name] ;
if (messageFault != null )
{

TreeNode treeNode=new TreeNode() ;

tnFault.ImageIndex=14;
tnFault.SelectedImageIndex=14;
if (treeNode != null)
tnFault.Nodes.Add (treeNode);
};

};
```

```
            if (tnFault.Nodes.Count > 0 ) tnOperation.Nodes.Add  (tnFault );

            return tnOperation;

            }

            public void Parse()
            {
            foreach (Service service in _services[0].Services )
            {
            TreeNode tnService=new TreeNode(service.Name) ;
            tnService.ImageIndex=1;
            tnService.SelectedImageIndex=1;

            foreach (Port port in service.Ports)
            {
            XmlQualifiedName bindName=port.Binding ;
            Binding bind=_services.GetBinding(bindName);
            PortType portType=_services.GetPortType(bind.Type) ;
        TreeNode tnPort=new TreeNode (port.Name);
            tnPort.ImageIndex=6;
            tnPort.SelectedImageIndex=6;
            string protocol=GetProtocol (bind);
            string portDesc="Protocol: " + protocol + "\n";
            switch (protocol)
            {
            case "Soap":
            {
            SoapAddressBinding ad=(SoapAddressBinding)port.Extensions.Find(typeof(SoapAddressBinding));
            portDesc += "Location: " + ad.Location + "\n";
            break;
             }
             case "HttpGet":
            {
            HttpAddressBinding ad=(HttpAddressBinding)port.Extensions.Find(typeof(HttpAddressBinding));
            portDesc += "Location: " + ad.Location + "\n";
break;
            case "HttpPost":
            {
            HttpAddressBinding ad=(HttpAddressBinding)port.Extensions.Find(typeof(HttpAddressBinding));
portDesc += "Location: "   + ad.Location + "\n";
```

```
break;
}
}
tnPort.Tag=portDesc;
foreach (OperationBinding obin in bind.Operations)
{
foreach (Operation oper in portType.Operations)
if (obin.Name.Equals(oper.Name) )
{
TreeNode tnOper=TranslateOperation (port, obin, oper, protocol);

tnOper.ImageIndex=11;
tnOper.SelectedImageIndex=11;
if (tnOper != null)
{
tnPort.Nodes.Add (tnOper);              };
}
}
tnPort.Expand ();
tnService.Nodes.Add(tnPort) ;
}
ServiceNode.Nodes.Add (tnService);
}
}
}
}
```

```
public class FacadeAcesso {
    consultarServicoWeb() {
       DocumentoWSDLInternetImpl docwsdl = new DocumentoWSDLInternetImpl();
       docwsdl. obterServicoWeb();
   }
}
public class DocumentoWSDLInternetImpl {
    obterServicoWeb() {
   }
```

```
}
public class OwlBLL {
    OwlDAL dao =new OWlDAL();
   public void  excluir(Ontologia ontologia) {
    dao.excluir(ontologia)
   }

    public void incluir(Ontologia ontologia) {
          dao.incluir(ontologia)
   }
}
public class OwlDAL {
    RepositorioOWL repositorioowl = new RepositorioOWL();
   public void excluir(Ontologia ontologia) {
      repositorioowl.excluir(ontologia);
   }
   public void incluir(Ontologia ontologia) {
      repositorioowl. incluir(ontologia);

   }
}
using System;
using System.Collections;
using System.Collections.Generic;
using System.Data;
using System.Diagnostics;
using System.Data.SqlClient;
using System.Runtime.Remoting.Messaging;
```

```
public class RepositorioOWL
{
	protected SqlCommand vrComando;
	#region "Atributo"
		#endregion
	protected string Lstring_conexao = "";
	#region " Construtor "
	public RepositorioOWL()
	{
		Lstring_conexao = Gcte_conexao_bd;
	}
	public RepositorioOWL(Pstring_conexao)
	{
		Lstring_conexao = Pstring_conexao;
	}
	#endregion
	#region "Constantes"
	public const string Gcte_conexao_bd = "StringConexao";
	public const string Gcte_conexao_bd_qtd_loop = "QtdMaxLoop";
	#endregion



	#region " GetSQLParameterType "
	private SqlDbType obter_tipos_parametros_sql(object Objeto)
	{
		SqlDbType vrDBType = SqlDbType.NVarChar;
		switch (Objeto.GetType().ToString()) {
			case "System.String":
				vrDBType = SqlDbType.NVarChar;
				break;
			case "System.Byte":
				vrDBType = SqlDbType.TinyInt;
				break;
			case "System.Int16":
```

```
                    vrDBType = SqlDbType.SmallInt;
                    break;
                case "System.Int32":
                    vrDBType = SqlDbType.Int;
                    break;
                case "System.Int64":
                    vrDBType = SqlDbType.BigInt;
                    break;
                case "System.Boolean":
                    vrDBType = SqlDbType.Bit;
                    break;
                case "System.DateTime":
                    vrDBType = SqlDbType.DateTime;
                    break;
                case "System.Decimal":
                    vrDBType = SqlDbType.Decimal;
                    break;
                case "System.Double":
                    vrDBType = SqlDbType.Float;
                    break;
                case "System.Single":
                    vrDBType = SqlDbType.Real;
                    break;
        }
        return vrDBType;
}
#endregion
#region " Parametro "
        public void limpar_parametro()
{
        vrComando.Parameters.Clear();
}
public void adicionar_parametro(object vrValor, string vrNome)
{
```

```
            SqlParameter vrParametro = obter_parametro(vrNome, obter_tipos_parametros_sql(vrValor), vrValor, ParameterDirection.Input, -1);
            vrComando.Parameters.Add(vrParametro);
        }
        public void adicionar_parametro(object vrValor, SqlDbType vrTipo, string vrNome)
        {
            SqlParameter vrParametro = obter_parametro(vrNome, vrTipo, vrValor, ParameterDirection.Input, -1);
            vrComando.Parameters.Add(vrParametro);
        }
        private SqlParameter obter_parametro(string vrNome, SqlDbType vrDataType, object vrValor, ParameterDirection vrDirecao, Int32 vrTamanho)
        {
            SqlParameter vrParam = new SqlParameter(vrNome, vrDataType);
            vrParam.Value = vrValor;
            vrParam.Direction = vrDirecao;

            if ((vrTamanho != -1)) {
                vrParam.Size = vrTamanho;
            }

            return vrParam;
        }
        #endregion
        #region " ExecutarSql "
        public Int32 executar_sql(string vrClausula)
        {
        //, Optional ByVal PboRecepcao As Boolean = False) As Int32
            SqlConnection Conexao = null;
            Int32 Lretorno = default(Int32);
            try {
                //Conexao = criar_conexao(PboRecepcao)
                Conexao = criar_conexao();
                preparar_comando(vrClausula, CommandType.Text, Conexao);
                Lretorno = vrComando.ExecuteNonQuery();
            } catch (Exception ex) {
```

```
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return Lretorno;
        }

        public Int32 executar_sql(string strProcedure, ArrayList Pparameters)
        {
            SqlConnection Conexao = null;
            Int32 Lretorno = default(Int32);
            try {
                Conexao = criar_conexao();
                preparar_comando(strProcedure, CommandType.StoredProcedure,
Pparameters, Conexao);
                Lretorno = vrComando.ExecuteNonQuery();
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return Lretorno;
        }
        #endregion
        #region " ExecutarDataSet "
        public DataSet obter_registro_dst(string strProcedure)
        {
            DataSet DSRetorno = new DataSet();
            SqlDataAdapter Adapter = null;
```

```
            SqlConnection Conexao = null;
            try {
                Conexao = criar_conexao();
                preparar_comando(strProcedure, CommandType.StoredProcedure,
Conexao);
                Adapter = new SqlDataAdapter(vrComando);
                Adapter.Fill(DSRetorno);
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return DSRetorno;
        }

        public DataSet obter_registro_dst(string strProcedure, ArrayList Pparameters)
        {
        DataSet DSRetorno = new DataSet();
            SqlDataAdapter Adapter = null;
            SqlConnection Conexao = null;
            try {

                Conexao = criar_conexao();
                preparar_comando(strProcedure, CommandType.StoredProcedure,
Pparameters, Conexao);
                Adapter = new SqlDataAdapter(vrComando);
                Adapter.Fill(DSRetorno);
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
```

```
            }
            return DSRetorno;
        }

        public bool obter_registro_bool(string vrClausula)
        {

            bool Retorno = false;
            SqlDataReader Adapter = null;
            SqlConnection Conexao = null;

            try {
                Conexao = criar_conexao();
                preparar_comando(vrClausula, CommandType.StoredProcedure,
Conexao);
                Adapter = vrComando.ExecuteReader();
                if ((Adapter != null)) {
                    Adapter.Read();
                    Retorno = Adapter.HasRows;
                    Adapter.Close();
                }
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return Retorno;
        }

        public bool obter_registro_bool(string vrClausula, ArrayList Pparameters)
        {

            bool Retorno = false;
            SqlDataReader Adapter = null;
            SqlConnection Conexao = null;
```

```
            try {
                Conexao = criar_conexao();
                preparar_comando(vrClausula, CommandType.StoredProcedure,
Pparameters, Conexao);
                Adapter = vrComando.ExecuteReader();
                if ((Adapter != null)) {
                    Adapter.Read();
                    Retorno = Adapter.HasRows;
                    Adapter.Close();
                }
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return Retorno;
        }

        public bool obter_registro_bool(string strPrcedure, CommandType PcmdType)
        {
            bool Retorno = false;
            SqlDataReader Adapter = null;
            SqlConnection Conexao = null;

            try {
                Conexao = criar_conexao();
                preparar_comando(strPrcedure, PcmdType, Conexao);
                Adapter = vrComando.ExecuteReader();
                if ((Adapter != null)) {
                    Adapter.Read();
                    Retorno = Adapter.HasRows;
                    Adapter.Close();
                }
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
```

```
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return Retorno;
        }

        public int obter_registro_sp(string vrClausula, CommandType PcmdType, ArrayList
Pparameters)
        {

            int Retorno = 0;
            SqlConnection Conexao = null;
            try {
                Conexao = criar_conexao();

                preparar_comando(vrClausula, PcmdType, Pparameters, Conexao);
                Retorno = vrComando.ExecuteScalar();
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return Retorno;
        }

        public DataSet obter_registro_sp_dst(string vrClausula, CommandType PcmdType,
ArrayList Pparameters)
        {

            DataSet DSRetorno = new DataSet();
            SqlDataAdapter Adapter = null;
            SqlConnection Conexao = null;
            try {
                Conexao = criar_conexao();

                preparar_comando(vrClausula, PcmdType, Pparameters, Conexao);
                Adapter = new SqlDataAdapter(vrComando);
                Adapter.Fill(DSRetorno);
            } catch (Exception ex) {
                throw ex;
            } finally {
                if ((Conexao != null)) {
```

```
                    Conexao.Close();
                }
                vrComando.Dispose();
            }
            return DSRetorno;
        }
        #endregion

        #region " PrepararComando "


        protected void preparar_comando(string clausulaSQL, CommandType cmdType, SqlConnection conexaoCorrente)
        {
            SqlTransaction objTransacao = null;
            vrComando = new SqlCommand();

            objTransacao = obter_transacao();

            if ((objTransacao != null)) {
                conexaoCorrente = objTransacao.Connection;
                vrComando.Transaction = objTransacao;
            }
            if ((conexaoCorrente == null)) {
                conexaoCorrente = criar_conexao();
            }
            vrComando.Connection = conexaoCorrente;
            vrComando.CommandText = clausulaSQL;
            vrComando.CommandType = cmdType;
            vrComando.CommandTimeout = conexaoCorrente.ConnectionTimeout;
        }
        protected void preparar_comando(string clausulaSQL, CommandType cmdType, ArrayList parameters, SqlConnection conexaoCorrente)
        {
            SqlTransaction objTransacao = null;
            vrComando = new SqlCommand();
```

```
            objTransacao = obter_transacao();
            if ((objTransacao != null)) {
                conexaoCorrente = objTransacao.Connection;
                vrComando.Transaction = objTransacao;
            }
            if ((conexaoCorrente == null)) {
                conexaoCorrente = criar_conexao();
            }
            vrComando.Connection = conexaoCorrente;
            vrComando.CommandText = clausulaSQL;
            vrComando.CommandTimeout = conexaoCorrente.ConnectionTimeout;
            SqlParameter param = null;
            foreach (SqlParameter param_loopVariable in parameters) {
                param = param_loopVariable;
                vrComando.Parameters.Add(param);
            }
            vrComando.CommandType = cmdType;
        }
        #endregion
        #region " Transação "
        protected SqlTransaction obter_transacao()
        {
            SqlTransaction transacao = (SqlTransaction)CallContext.GetData("transacao"
+ Lstring_conexao);
            return transacao;
        }
        public static void comecar_transacao(string Ptipo_conexao)
        {
            IDbTransaction objTransacao =
(IDbTransaction)CallContext.GetData("transacao" + Ptipo_conexao);
            Int32 contador = default(Int32);
            if ((objTransacao == null)) {
                SqlConnection conexao = criar_conexao(Ptipo_conexao);
                //, PboRecepcao)
                objTransacao =
conexao.BeginTransaction(IsolationLevel.ReadCommitted);
                CallContext.SetData("transacao" + Ptipo_conexao, objTransacao);
                CallContext.SetData("contadorTransacao" + Ptipo_conexao, 1);
```

```
                    CallContext.SetData("commitTransacao" + Ptipo_conexao, true);
                } else {
                    if ((CallContext.GetData("contadorTransacao" + Ptipo_conexao) != null)) {
                        contador = Convert.ToInt32(CallContext.GetData("contadorTransacao" + Ptipo_conexao));
                        contador = contador + 1;
                        CallContext.SetData("contadorTransacao" + Ptipo_conexao, contador);
                    }
                }
            }

            public static void confirmar_transacao(string Ptipo_conexao)
            {
                IDbConnection conexaoCorrente = null;
                IDbTransaction transacaoCorrente = (IDbTransaction)CallContext.GetData("transacao" + Ptipo_conexao);
                bool fecharConexao = false;
                Int32 contador = default(Int32);
                bool commit = false;
                if ((transacaoCorrente != null)) {
            try {
            if (((CallContext.GetData("contadorTransacao" + Ptipo_conexao) != null) & (CallContext.GetData("commitTransacao" + Ptipo_conexao) != null))) {
            contador = Convert.ToInt32(CallContext.GetData("contadorTransacao" + Ptipo_conexao));
            commit = Convert.ToBoolean(CallContext.GetData("commitTransacao" + Ptipo_conexao));
            if ((contador == 1)) {
            fecharConexao = true;
            conexaoCorrente = transacaoCorrente.Connection;
            try {
            if ((commit)) {
            transacaoCorrente.Commit();
            } else {
            transacaoCorrente.Rollback();
```

```
        }
        } finally {
        CallContext.FreeNamedDataSlot("transacao" + Ptipo_conexao);
        CallContext.FreeNamedDataSlot("contadorTransacao" + Ptipo_conexao);
        CallContext.FreeNamedDataSlot("commitTransacao" + Ptipo_conexao);
        }
        } else {
        contador = contador - 1;
        CallContext.SetData("contadorTransacao" + Ptipo_conexao, contador);
        }
        } else {
       throw new Exception("Contexto de transação contém informações incompletas");
        }
      } finally {
        if ((fecharConexao)) {
        if ((conexaoCorrente != null)) {
        if ((conexaoCorrente.State != ConnectionState.Closed)) {
        conexaoCorrente.Close();
        }
        }}       }}
        }
        public static void desfazer_transacao(string Ptipo_conexao)
        {
                IDbConnection conexaoCorrente = null;
                bool fecharConexao = false;
                IDbTransaction transacaoCorrente =
(IDbTransaction)CallContext.GetData("transacao" + Ptipo_conexao);
                Int32 contador = default(Int32);
                if ((transacaoCorrente != null)) {
                try {
        if (((CallContext.GetData("contadorTransacao" + Ptipo_conexao) != null) &
        (CallContext.GetData("commitTransacao" + Ptipo_conexao) != null))) {
                contador = Convert.ToInt32(CallContext.GetData("contadorTransacao" +
Ptipo_conexao));
                CallContext.SetData("commitTransacao" + Ptipo_conexao, false);
                if ((contador == 1)) {
```

```
                    fecharConexao = true;
                    conexaoCorrente = transacaoCorrente.Connection;
                    try {
                    transacaoCorrente.Rollback();
                    } finally {
                    CallContext.FreeNamedDataSlot("transacao" + Ptipo_conexao);
                      CallContext.FreeNamedDataSlot("contadorTransacao"                    +
Ptipo_conexao);

        CallContext.FreeNamedDataSlot("commitTransacao" + Ptipo_conexao);
        }
        } else {
        contador = contador - 1;
        CallContext.SetData("contadorTransacao" + Ptipo_conexao, contador);
        }
        } else {
        throw new Exception("Contexto de transação contém informações incompletas");
        }
     } finally {
    if ((fecharConexao)) {
        if ((conexaoCorrente != null)) {
        if ((conexaoCorrente.State != ConnectionState.Closed)) {
        conexaoCorrente.Close();
        }
        }         }
        }           }
                  }
        }
        #endregion
        #region "Criar Parâmetro"
        /// <summary>
        /// Criar o parâmetro para instrução SQL.
        /// </summary>
        /// <param name="Pnome"></param>
        /// <param name="Ptipo"></param>
        /// <param name="Pvalor"></param>
        /// <returns></returns>
        /// <remarks></remarks>
        public SqlParameter criar_parametro(string Pnome, SqlDbType Ptipo, object Pvalor,
        bool     PboStringVazia = false)
        {
                SqlParameter param = new SqlParameter();
```

```
            param.ParameterName = Pnome;
            param.SqlDbType = Ptipo;
            if (Pvalor == null) {
                param.Value = DBNull.Value;
            } else {
            if ((Ptipo == SqlDbType.VarChar & Pvalor.ToString().Length == 0) &
!PboStringVazia) {
            param.Value = DBNull.Value;
                } else {
                    param.Value = Pvalor;
                }
            }
            return param;
        }
        #endregion

        #region " Conexão "

        protected virtual SqlConnection criar_conexao()

        {

            SqlConnection Fcnn = null;

        try {

        System.Configuration.AppSettingsReader app_reader = new
        System.Configuration.AppSettingsReader();

        string str_conexao = Convert.ToString(app_reader.GetValue(Lstring_conexao,
        typeof(string)));

        Fcnn = new SqlConnection(str_conexao);

        Fcnn.Open();

        return Fcnn;

        } catch (SqlException ex) {

        throw ex;}}

        protected static SqlConnection criar_conexao(string Ptipo_conexao)

        SqlConnection Fcnn = null;

        try {

        System.Configuration.AppSettingsReader app_reader = new
        System.Configuration.AppSettingsReader();

        string str_conexao = Convert.ToString(app_reader.GetValue(Ptipo_conexao,
typeof(string)));

        Fcnn = new SqlConnection(str_conexao);

        Fcnn.Open();

        return Fcnn;

        } catch (SqlException ex) {

        throw ex;
```

```
        }
        }
        #endregion
}
```

ANEXO A - CONTRATO WSDL (LIVRARIA.WSDL) DESCREVENDO O *WEB SERVICE RESTFUL* ANOTADO SEMANTICAMENTE

```
<wsdl:description xmlns:wsdl="http://www.w3.org/ns/wsdl"
    targetNamespace="http://www.bookstore.org/booklist/wsdl"
    xmlns:tns="http://www.bookstore.org/booklist/wsdl"
    xmlns:whttp="http://www.w3.org/ns/wsdl/http"
    xmlns:wsdlx="http://www.w3.org/ns/wsdl-extensions">
  <wsdl:interface name="BookListInterface">
    <wsdl:operation name="getBookList"
       pattern="http://www.w3.org/ns/wsdl/in-out"
       style="http://www.w3.org/ns/wsdl/style/iri"
       wsdlx:safe="true">
      <wsdl:input element=""/>
      <wsdl:output element=""/>
    </wsdl:operation>
  </wsdl:interface>
  <wsdl:binding name="BookListHTTPBinding"
     type="http://www.w3.org/ns/wsdl/http"
     interface="tns:BookListInterface">
    <wsdl:documentation>
       The RESTful HTTP binding for the book list service.
    </wsdl:documentation>
    <wsdl:operation ref="tns:getBookList" whttp:method="GET"/>
  </wsdl:binding>
  <wsdl:service name="BookList" interface="tns:BookListInterface">
    <wsdl:documentation>
       The bookstore's book list service.
    </wsdl:documentation>
    <wsdl:endpoint name="BookListHTTPEndpoint"
       binding="tns:BookListHTTPBinding"
       address="http://www.bookstore.com/books/">
    </wsdl:endpoint>
  </wsdl:service>
</wsdl:description>
```