Universidad Autónoma Metropolitana

Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería

Extracción automatizada y representación de servicios Web mediante ontologías

Manual de usuario

Alumno
Jorge Pascual Martínez
207202192

Asesora:
Dra. Maricela Claudia Bravo Contreras
Departamento de Sistemas

Diciembre 2012

Índice

1. Interfaz del sistema

Para poder acceder a la aplicación se debe ingresar a la siguiente url `http://gisi.uam.azc.mx/sdws/descripcion-semantica-de-servicios-web`. La Figura 1 nos muestra la interfaz inicial del sistema.

Figura 1: Interfaz inicial del sistema

En la Figura 2 se observan los lenguajes de descripción de servicios (SDL) con los que la aplicación trabaja.Podemos seleccionar cualquiera de las tres opciones.

Para la elaboración de este manual se utilizó la primera opción que corresponde a WSDL 1.X.

Figura 2: SDLs disponibles

2. Seleccionar SDL

Una vez seleccionado la opción con la que vamos a trabajar, hacemos clic en el botón iniciar, esto nos mostrara el componente fileUploader (ver Figura 3) para poder cargar los archivos de descripción de servicios, en este caso nos permitirá cargar archivos WSDL.

Cuando se trabaja con mas de 200 archivos, se sugiere que la subida de archivos se realice en bloques de no más de 200 archivos. esto con el fin de evitar que nuestro explorador deje de responder.

Figura 3: Componente fileUploader para cargar archivos

3. Subir archivos al servidor

Para buscar los archivos que se van a utilizar presionamos el botón buscar, esta acción nos abrirá el explorador de archivos, de tal manera que podamos navegar hasta la ruta donde se encuentre la colección, como se muestra en la Figura 4, posteriormente seleccionamos el o los archivos que se subirán al sistema y damos clic en el botón abrir.

Una vez que presionamos el botón abrir se nos mostraran en el componente los archivos seleccionados. Y procedemos a subir los archivos al servidor, esto se realiza al presionar el botón subir.

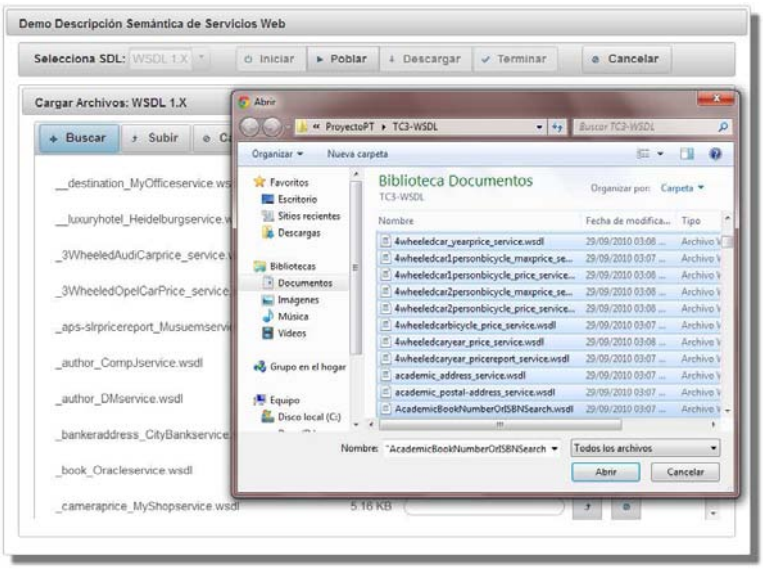

Figura 4: Buscar y seleccionar archivos

La Figura 5 muestra los errores que se producen cuando se intenta subir archivos con extensión distinta al SDL que hayamos seleccionado. De tal manera que no se permite la carga del archivo al servidor.

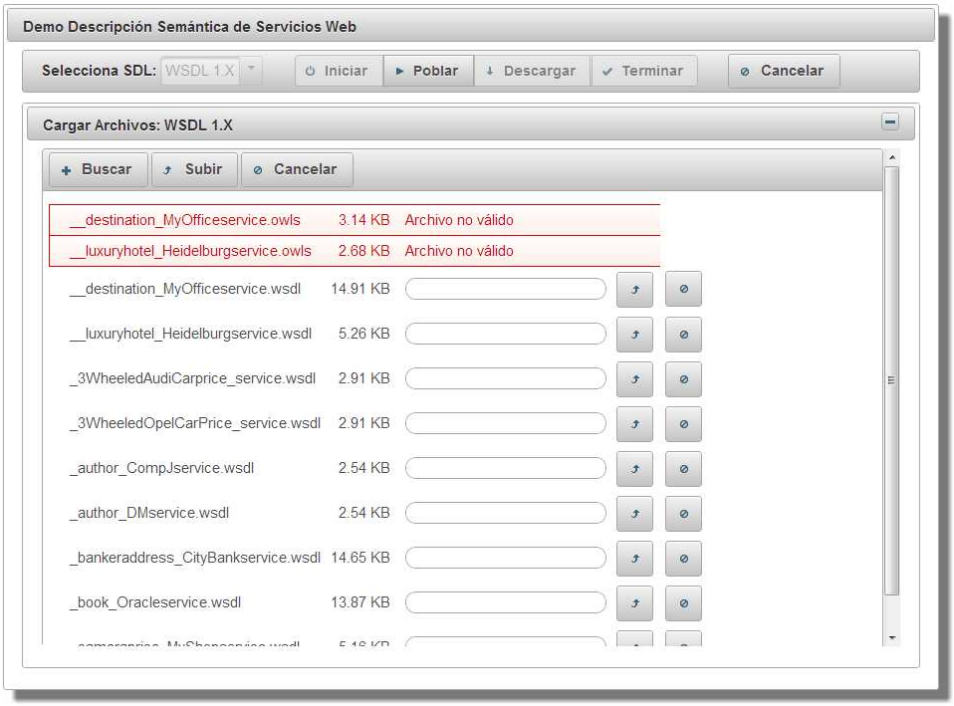

Figura 5: Selección de archivos con extensión invalida

4. Finalizar la carga de archivos

La Figura 6 nos muestra una lista con los nombres de los archivos que se subieron al servidor y el total de ellos. Si se requiere subir más archivos volvemos a realizar los pasos anteriores, es decir, buscar, seleccionar y subir los archivos.

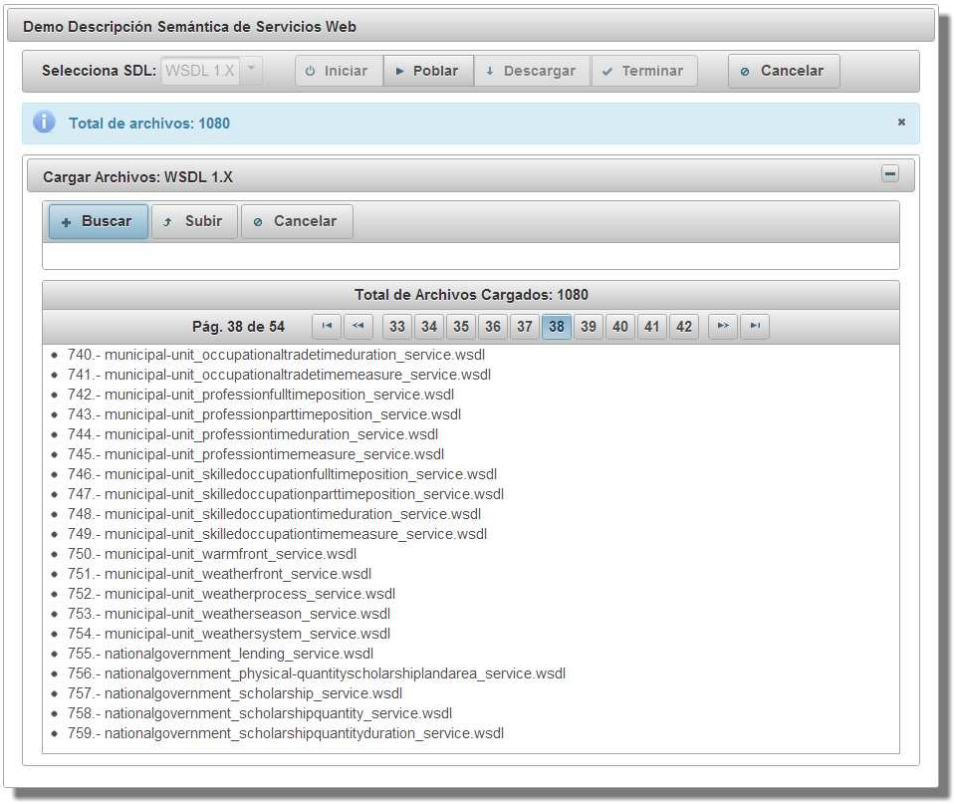

Figura 6: Total de archivos cargados

5. Iniciar parser y poblado

Una vez realizado la carga de archivos procedemos a poblar la ontología, para iniciar este proceso damos clic en el botón poblar. El proceso de poblado implica la extracción de información de cada uno de los archivos de descripción, la inserción de esta información en el modelo ontológico y guardar los cambios que se realizan en el modelo.

En la Figura 7 se observa el estado de la interfaz después de hacer clic en el botón poblar, se nos muestra una ventana modal que nos indica que el sistema está trabajando, el tiempo que tarda la interfaz en terminar el proceso, depende del número de archivos con los que se trabaje.

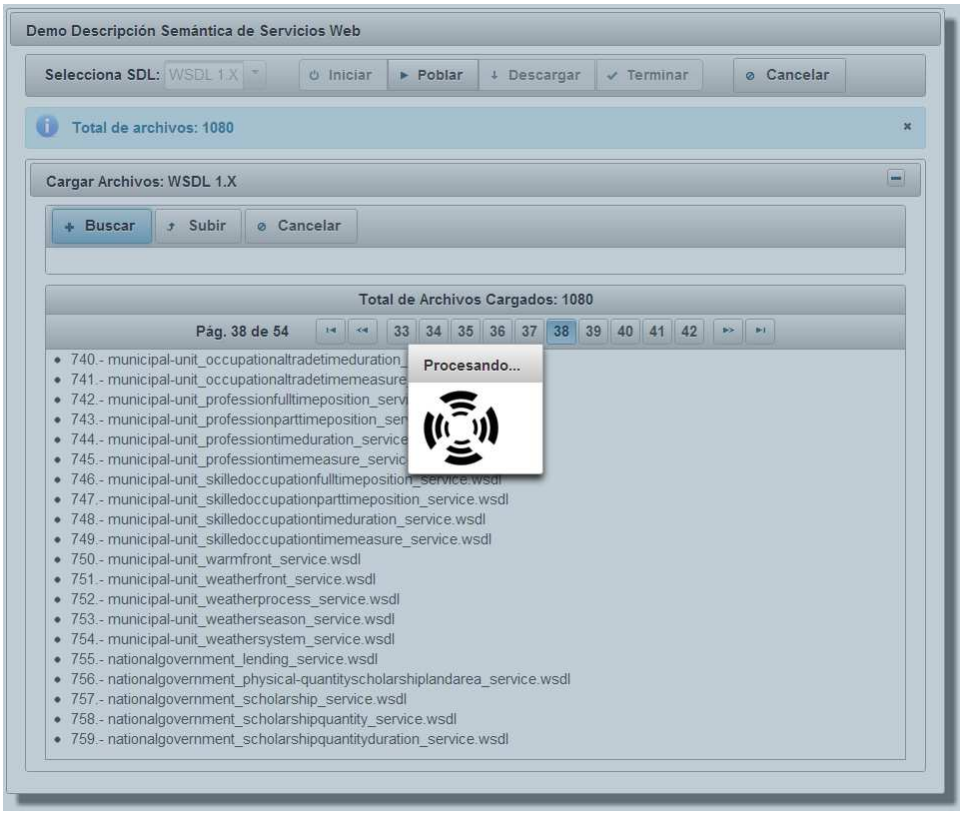

Figura 7: Estado de la interfaz al realizar el poblado

6. Proceso de poblado terminado

La Figura 8 nos muestra la interfaz después de terminar el poblado, se puede observar que se han habilitado los botones Descargar y Terminar, el botón Descargar nos permite guardar en nuestra computadora el archivo OWL que contiene toda la información que se recupero de los archivos de descripción. Finalmente damos clic en el botón Terminar para volver al estado inicial de la interfaz y poder trabajar con otro SDL.

Si ocurrió algún error durante el procesamiento de los archivos la aplicación mostrara los errores que se produjeron, y una descripción del mismo tal como se muestra en la Figura 9

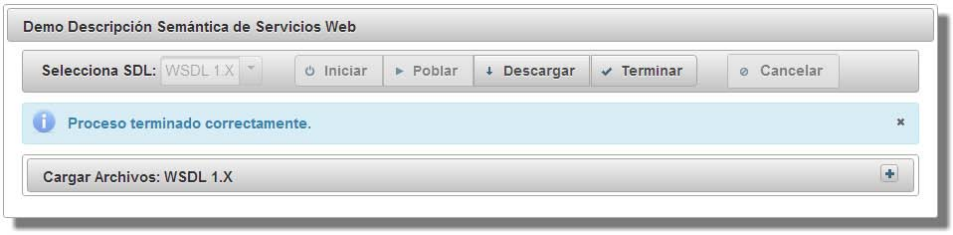

Figura 8: Estado de la interfaz al terminar el proceso de poblado correctamente

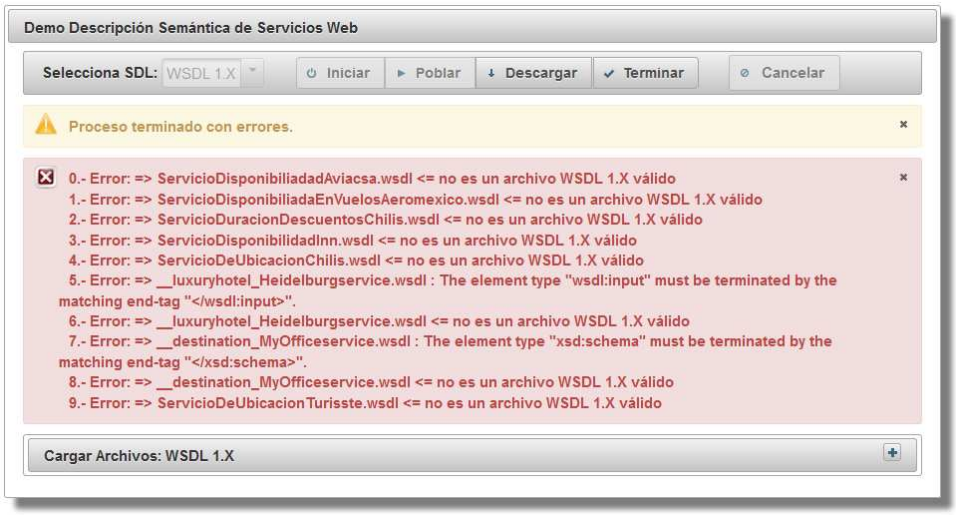

Figura 9: Estado de la interfaz al terminar el proceso con errores

7. Descargar ontologia

En la Figura 10 se observa el archivo descargado.

Figura 10: Archivo OWL descargado

8. Abrir ontologia con Protégé

Después de haber realizado la descarga de la ontología procedemos a abrir el archivo, Protégé 4.1 nos permite trabajar con ontologías, por lo que utilizamos este programa para abrir el archivo OWL. La Figura 11 muestra la interfaz inicial de Protégé , seleccionamos la opción open OWL ontology, y procedemos a buscar el archivo que hemos descargado anteriormente y damos clic en abrir.

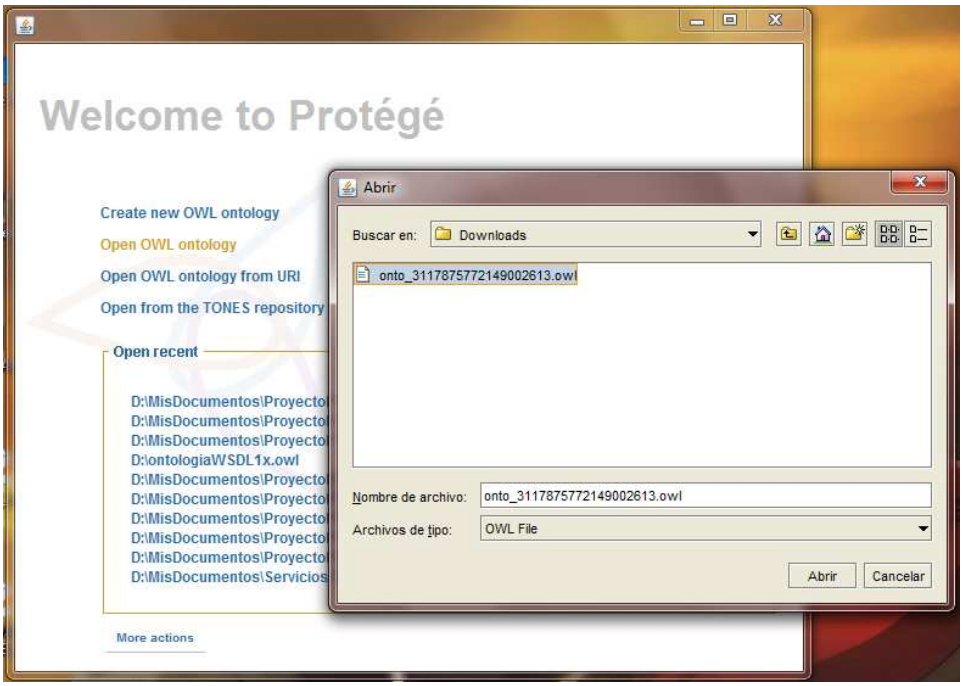

Figura 11: Abrir ontología con Protégé

9. Vista de las entidades con Protégé

La Figura 12 nos muestra la vista Entities de la interfaz de Protégé después de haber abierto el archivo, podemos observar las clases que conforman la ontología así como los individuos que existen por cada clase.

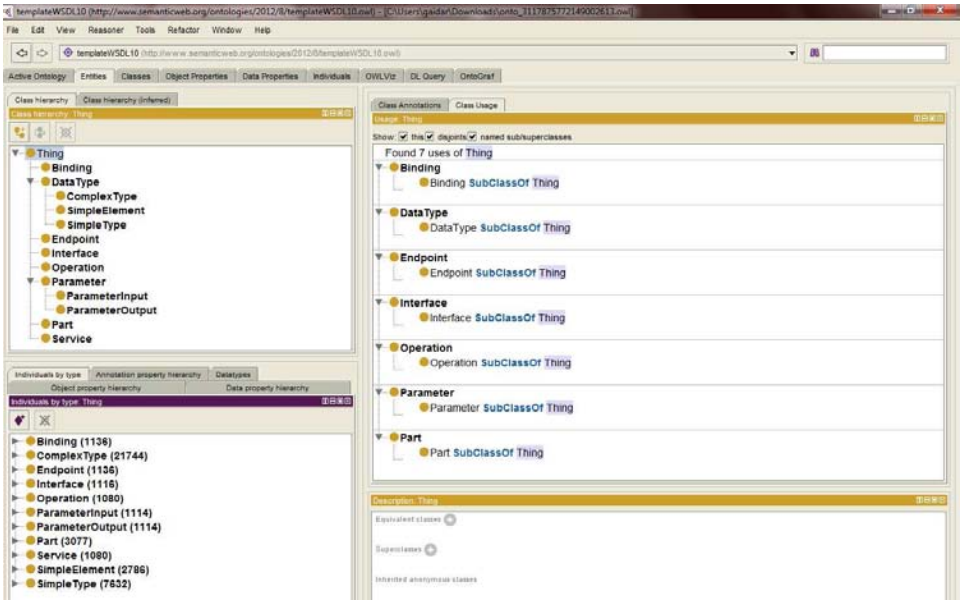

Figura 12: Vista Entities en Protégé

10. Vista de los individuos con Protégé

En la Figura 13 se observa la vista Individuals, esta vista nos muestra el nombre de los individuos de la clase que se haya seleccionado, es decir, nos muestra los miembros que pertenecen a la clase seleccionada. También nos proporciona toda la información asociada a un individuo que se seleccione.

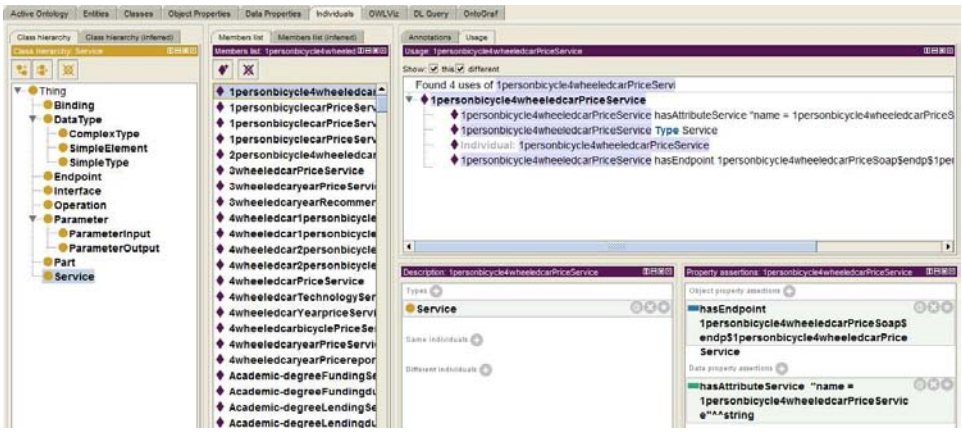

Figura 13: Vista Individuals en Protégé

11. Vista OntoGraf con Protégé

En la Figura 14 se muestra la vista OntoGraf que nos proporciona Protégé, en esta imagen se aprecia de forma grafica las relaciones que existen entre las clases y los individuos que conforman la ontología, del lado derecho se muestra las relaciones que representa cada una de las flechas dentro del grafo.

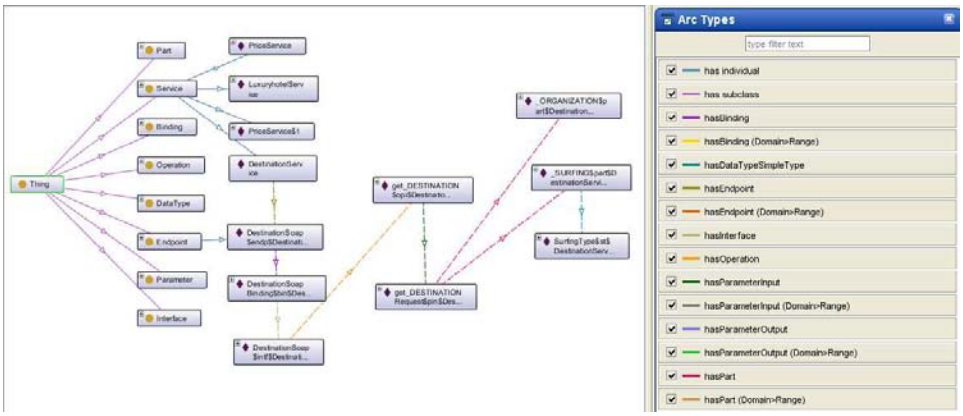

Figura 14: Vista OntoGraf en Protégé

Para poder utilizar las opciones WSDL 2.0 y OWLS se procede de la misma manera, es decir, se siguen los mismos pasos que se usaron para trabajar con WSDL 1.X.

UNIVERSIDAD AUTÓNOMA METROPOLITANA

UNIDAD AZCAPOTZALCO

DIVISIÓN DE CIENCIAS BÁSICAS E INGENIERÍA

Extracción automatizada y representación de servicios Web mediante ontologías

REPORTE FINAL DE PROYECTO TERMINAL

Alumno
Jorge Pascual Martínez
207202192

Asesora:
Dra. Maricela Claudia Bravo Contreras
Departamento de Sistemas

DICIEMBRE 2012

Índice

1. Introducción

Actualmente el internet se ha convertido en una medio fundamental para que las empresas den a conocer a sus clientes los servicios que proporcionan. Permitiendo con ello penetrar a un mayor mercado de clientes potenciales y obtener mayores beneficios para el negocio.

Sin embargo, las aplicaciones de negocios de las empresas cada vez son más complejas, ya no solo se limitan a proporcionar servicios sobre ellas sino que ofrecen al usuario servicios de terceros, permitiendo con ello darle un valor agregado a su aplicación, por ejemplo una empresa que ofrece servicios de viajes debe proporcionar al cliente la información relacionada con el hotel y los vuelos, los dos negocios son independientes uno del otro, sin embargo, deben de suministrar la información requerida por el usuario.

Para poder ofrecer la colaboración entre las distintas aplicaciones que necesiten interactuar entre ellas surgen las aplicaciones SOA[1], permitiendo el desarrollo de aplicaciones compuestas de servicios que se encuentran publicados en muchos sistemas remotos. La principal metodología de desarrollo de las aplicaciones SOA es mediante el uso de servicios Web.

Los servicios Web de acuerdo con la W3C [2] son sistemas de software diseñados para interoperar en la Web. Estas aplicaciones intercambian datos entre sí con el objetivo de ofrecer sus servicios. La interacción entre el servicio Web y un cliente se realiza a través de mensajes SOAP[3], transportando los mensajes usando HTTP[4] con una serialización XML[5] conjuntamente con otros estándares web. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Para describir la interfaz pública de los servicios Web se utilizan distintos SDL[6], lo que ocasiona una gran dificultad para la búsqueda e invocación automatizada de servicios Web. Estos lenguajes de descripción utilizan una estructura distinta, sin embargo, contienen elementos comunes, los cuales proporcionan la interfaz de comunicación que utiliza el cliente para crear un proxy que invoque al servicio de forma remota. Los desarrolladores de software a menudo buscan en los repositorios de servicios Web públicos aquellos que cumplan con ciertos requerimientos para integrarlos a sus aplicaciones. Sin embargo, la implementación de las interconexiones de los servicios Web debe realizarse de forma manual o semiautomática.

Una característica de las aplicaciones compuestas por servicios Web, es que éstas pueden estar dispersas por la red, escritos en diferentes lenguajes de programación y en ambientes de ejecución diferentes, sin que esto impida que puedan

[1] Arquitectura orientada a servicios (*Service Oriented Architecture*).
[2] Consorcio World Wide Web (*World Wide Web Consortium)*)
[3] Protocolo simple de acceso a objetos (*Simple Object Access Protocol)*
[4] Protocolo de transferencia de hipertexto *(Hypertext Transfer Protocol)*
[5] Lenguaje de marcado extensible (*eXtensible Markup Language)*
[6] Lenguajes de descripción de servicios *(Service Description Language)*

interactuar entre sí. De esta manera, la interoperabilidad entre los diferentes componentes de un sistema, e incluso entre aplicaciones, se favorece.

Como parte de la infraestructura para la utilización de los servicios Web, existen diversos directorios en los cuales se deposita la información de los servicios permitiendo así el desarrollo de aplicaciones con funcionalidades desarrolladas por terceros,generalmente los datos de los servicios web se encuentran publicados en repositorios que contienen información acerca de los servicios, el catálogo UDDI[7], permite realizar búsquedas más específicas de acuerdo a la categorización de las empresas y los servicios que esta ofrece. UDDI solo es un registro ya que no contendrá los archivos de descripción de los servicios pero si nos dirá donde podemos encontrarlo..

El proceso de descubrimiento de los servicios es tedioso para los desarrolladores y además tiene un fuerte impacto en el tiempo de desarrollo de software, el tiempo invertido en la búsqueda de servicios que resuelvan una problemática específica ocasionan costos lo que representa un gran problema en el desarrollo de los sistemas.

Además la falta de una buena documentación de los servicios, hace más complicado su invocación, para mejorar la documentación se han propuestos diferentes soluciones, una de ellas se basa en la Web semántica y el uso de ontologías.

Un estudio realizado a los siguientes lenguajes de descripción: WSDL[8], OWL-S[9], de acuerdo con [1], se obtuvo como resultado que la descripción funcional de los servicios Web se divide en dos partes, la transformación de la información realizada por el servicio y el cambio de estado como consecuencia de la ejecución del servicio. La transformación de la información se correlaciona con los parámetros de entrada y salida, mientras que el cambio de estado describe una condición de modificación después de la ejecución del servicio.

		WSDL	**OWLS**
Descripcion funcional		Operación	Perfil: nombre del servicio
Transformación de la información	Entrada	Entrada	Perfil:Tiene entrada
	Salida	Salida	Perfil:Tiene salidas
Cambio de estado	Estado anterior		Perfil:tiene precondiciones
	Estado posterior		Perfil:Tiene resultados

Cuadro 1: Descripción funcional de SDLs.

Usando la correlación que se describe en el Cuadro 1, entre estos modelos de descripción de servicios, se crearon modelos para representar a los servicios Web. Los modelos fueron representados mediante el diseño de ontologias, se utilizo OWL[10]

[7] *Universal Description, Discovery and Integration*

[8] Lenguaje de descripción de servicios Web *(Web Service Description Language)*

[9] Marcado semántico para servicios Web *(Semantic Markup for Web Services)*

[10] Lenguaje de ontologías Web *(Web Ontology Language)*

como lenguaje para definir los modelos ontológicos, la motivación acerca del uso de OWL [2] es por ser una recomendación de la W3C y por ser actualmente el lenguaje más utilizado para la creación de ontologías.

Este proyecto toma como punto de partida los resultados expuestos anteriormente e implementa una aplicación que dado un archivo de descripción de un servicio Web escrito en WSDL[3] [4] (versión 1.x y 2.0) y OWL-S [5] es representado en un modelo ontológico escrito en OWL.

Derivado de problemas anteriormente expuestos la relevancia de este proyecto es que colabora en la solución de buscar servicios Web de forma automatizada, ya que se creó una aplicación que permite que a partir de distintos archivos de descripción escritos en diferentes lenguajes de descripción de servicios, encontrar los elementos más relevantes y representarlos en un modelo ontológico.

Los beneficios de utilizar una ontología como modelo de representación son los siguientes:

- Definen de forma estándar los términos y sus relaciones dentro de un determinado dominio (área de conocimiento)
- Permiten el uso de restricciones y reglas de inferencia para hacer entender a las máquinas los conceptos que se manejan en ese dominio.
- Provee una forma para codificar el conocimiento y la semántica de tal manera que las máquinas puedan entenderlas

Los objetivos que se platearon en el proyecto fueron los siguientes:

Objetivo General:

- Diseñar e implementar un sistema que permita la extracción de la información más relevante de múltiples archivos de descripción de servicios Web y que sean representados mediante modelos ontológicos.

Objetivos Específicos:

- Diseñar un modelo ontológico para la representación de los servicios Web, considerando las diferencias sintácticas y de implementación de los lenguajes de descripción de servicios.
- Diseñar e implementar los módulos correspondientes a la interfaz de conectividad entre el sistema y los modelos ontológicos.
- Diseñar e implementar el modulo de extracción de información de WSDL 1.X y representación en OWL.
- Diseñar e implementar el modulo de extracción de información de WSDL 1.2 y representación en OWL

- Diseñar e implementar el modulo de extracción de información de OWLS y representación en OWL
- Diseño e implementación de la interfaz de usuario para el uso del sistema.

Servicios Web

Un servicio Web es un componente de software identificado por una URI [11], el cual puede ser accedido mediante protocolos y estándares basados en Internet utilizando la descripción de su interfaz de comunicación descrita en un formato procesable por una máquina, por ejemplo el lenguaje WSDL. El Servicio Web está diseñado para soportar interoperabilidad entre sistemas sobre una red. Otros sistemas interactúan con los Servicios Web de la manera descrita en su descripción usando mensajes SOAP, típicamente enviados usando HTTP con una serialización XML .

Una característica de las aplicaciones compuestas por Servicios Web, es que éstas pueden estar dispersas por la red, escritos en diferentes lenguajes de programación y en ambientes de ejecución diferentes, sin que esto impida que puedan interactuar entre sí. De esta manera, la interoperabilidad entre los diferentes componentes de un sistema, e incluso entre aplicaciones, se favorece.

Servicio Web semánticos

Los servicios web semánticos surgen de la necesidad de realizar las operaciones de descubrimiento, selección, composición, negociación, invocación, monitorización y recuperación semiautomática de los servicios web tradicionales.

Un servicio web semántico está formado por un servicio web y una anotación semántica sobre dicho servicio. La anotación semántica consiste en asociar conceptos y relaciones de una ontología con parámetros y operaciones de un servicio web.

Los Servicio Web Semánticos son servicios que se describen a sí mismos y son susceptibles al descubrimiento, composición e invocación automática. Dicho de otra manera, los servicios Web semánticos están habilitados con una descripción de sí mismos más formal, lo cual permite que sean interpretados por otros programas.

La descripción semántica de los Servicios Web va a permitir que las tareas de composición e invocación de servicios, entre diferentes usuarios y dominios sean automáticas. Existen tecnología para Servicios Web que solo proveen una descripción a nivel sintáctico, lo que provoca que tanto consumidores como proveedores tengan dificultad para representar el verdadero significado de los parámetros de entrada y salida de los servicios. Estas limitaciones pueden ser superadas mediante la representación semántica de los servicios mediante la utilización de una ontología de servicios, habilitando a las maquinas interpretar sus capacidades y hacer la integración con el respectivo dominio de conocimiento al que pertenece cada servicios.

[11] Identificador uniforme de recursos *(Uniform Resource Identifier)*

La Web semántica

> La Web semántica es una visión del futuro de la Web donde la información está dando un significado explícito, permitiendo que las máquinas puedan procesar automáticamente e integrar la información disponible en la Web. La Web semántica se basará en la capacidad de XML para definir esquemas de etiquetas a medida y en la aproximación flexible de RDF para representar datos. El primer nivel requerido por encima de RDF para la Web semántica es un lenguaje de ontologías que pueda describir formalmente el significado de la terminología usada en los documentos Web. Si se espera que las máquinas hagan tareas útiles de razonamiento sobre estos documentos, el lenguaje debe ir más allá de las semánticas básicas del RDF Schema.

La web semántica es una extensión de la web tradicional que permite a las máquinas interpretar el significado de los datos, en base a anotaciones semánticas basadas en descripciones ontológicas. Esta anotación semántica permite dotar a la gran cantidad de información de la web de una estructura lógica, basada generalmente en soluciones de gestión del conocimiento, que permite a los usuarios y las máquinas buscar y relacionar información de una forma más automática y directa.

Para anotar estos recursos de la web se emplean ontologías. Una ontología es una descripción de una pequeña parte del mundo real, de los tipos de elementos que aparecen en ese mundo, de las relaciones entre ellos, de los elementos existentes y de sus restricciones.

Ontologías

Una ontología se define como una especificación formal de una conceptualización compartida. En otras palabras, define los términos y relaciones básicas que comprenden el vocabulario de un tópico de área, así como las reglas para combinar los términos y relaciones para definir extensiones al vocabulario. Además, puede verse a las ontologías como un sistema para la representación de una base de conocimiento.

Las ontologías se han desarrollado dentro de la comunidad de investigación de Modelado del Conocimiento, con el fin de facilitar el intercambio y reutilización del conocimiento. Estas proveen una gran capacidad de expresividad al momento de hacer el modelado del dominio de conocimiento, lo que permite que éste pueda ser comunicado entre personas y sistemas de aplicación heterogéneos y distribuidos.

Al igual que con la arquitectura de servicios Web, la Web Semántica establece estándares los cuales tiene su base en URIs y XML Schema. Los actuales componentes del marco de trabajo de la Web Semántica son: RDF, RDF-Schema, y el lenguaje OWL.

OWL

Una ontología en OWL se puede definir básicamente mediante tres elementos principales:

- Las clases representan entidades del entorno que se esté describiendo,
- Las relaciones representan vínculos entre las entidades,
- Las instancias son afirmaciones concretas sobre el mundo real.

OWL define además un conjunto de propiedades que pueden ser aplicadas a los elementos anteriores, como pueden ser: restricciones de cardinalidad para las relaciones, propiedades de herencia para las clases, de igualdad o desigualdad, etc.

OWL ha sido diseñado para poder construir la Web semántica ya que se necesita poder representar el conocimiento de forma que sea legible por los ordenadores, esté consensuado, y sea reutilizable. OWL forma parte de un conjunto creciente de recomendaciones del W3C relacionadas con la Web semántica.

- XML proporciona una sintaxis superficial para documentos estructurados, pero no impone restricciones semánticas en el significado de estos documentos.
- XML Schema es un lenguaje que se utiliza para restringir la estructura de los documentos XML, además de para ampliar XML con tipos de datos.
- RDF es un modelo de datos para objetos (recursos") y relaciones entre ellos, proporcionando una semántica simple para éste. Este tipo de modelo de datos puede ser representado en una sintaxis XML.
- RDF Schema es un vocabulario utilizado para describir propiedades y clases de recursos RDF, con una semántica para la generalización y jerarquización tanto de propiedades como de clases.
- OWL añade más vocabulario para describir propiedades y clases: entre otros, relaciones entre clases, cardinalidad, igualdad, más tipos de propiedades, características de propiedades, y clases enumeradas.

Trabajos Relacionados

En la Tabla 2 se muestran los trabajos relacionados con el proyecto así como las similitudes y diferencias.

2. Desarrollo

2.1. Diseño

2.1.1. Arquitectura general del sistema

A continuación se muestra el diseño del sistema, así como el de las ontologías que se utilizaron en el proyecto. La Figura 1 muestra la arquitectura general del

Titulo	Similitudes	Diferencias
Plataforma de archivos de acceso remoto mediante servicios Web [6]	El trabajo se fundamenta en el uso de servicios Web	1. No es una aplicación que pretenda resolver los problemas relacionados con la invocación de servicios Web
B.Jeong, H. Cho, C.Lee [1]	Estudio de similitudes y comparación de SDLs. Planteamiento de un modelo general	1. No usa características funcionales 2. No propone un diseño concreto del modelo.
Una comparación conceptual entre WSMO y OWL-S [7]	Analiza y compara las similitudes de dos SDLs, plantea la correlación que existe entre ellos.	1. No realiza ninguna implementación funcional 2. Se limita a OWL-S 3. No considera servicios descritos en WSDL

Cuadro 2: Trabajos relacionados.

sistema, la aplicación recibe una colección de archivos de descripción de servicios Web escritos en alguno de los siguientes lenguajes de descripción.

1. WSDL 1.X
2. WSDL 2.0
3. OWLS

Posteriormente se procede a la extracción de los datos y al poblado del modelo ontológico.

2.1.2. Extracción y representación de WSDL 1.1

La Figura 2 muestra la estructura general de los archivos WSDL 1.1, a partir del ellos se obtendrá la información más importante contenida en cada uno de las secciones del archivo.

Cada sección del WSDL tiene su correspondencia con una clase en el modelo ontológico. La sección portType del WSDL se corresponde con la clase Interface en el modelo, esto es con el fin de homologar el nombre con WSDL 2.0. De igual

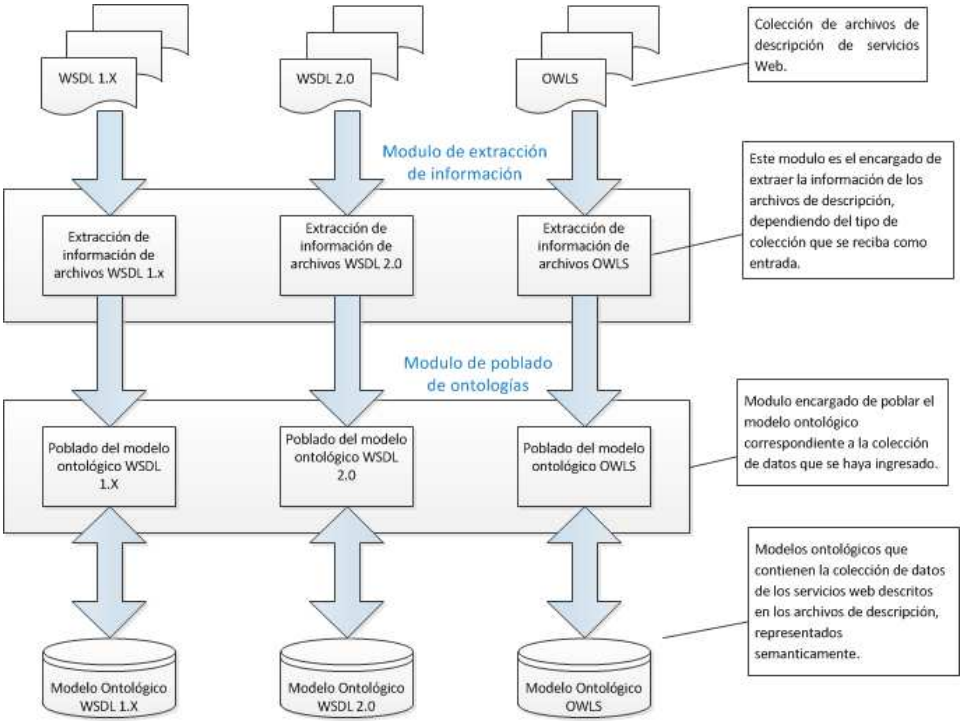

Figura 1: Arquitectura general del sistema

manera el port del wsdl 1.1 se corresponde con un endpoint en wsdl 2.0 y con la clase Endpoint en la ontología.

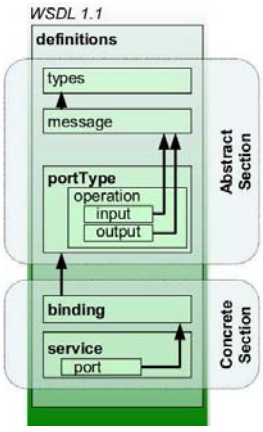

Figura 2: Estructura de un archivo WSDL 1.1

Partiendo de la estructura general de los archivos de descripción escritos en WSDL 1.1 se creó el modelo ontológico para representar a los servicios descritos

en este lenguaje de descripción. La Figura 3 muestra el diseño del modelo ontológico para WSDL 1.1, todas las clases derivan de una clase principal denominada owl: Thing, una clase es una descripción formal de una entidad del dominio que se quiere representar. Una clase puede tener subclases que representan conceptos que son más específicos que dicha clase. Por ejemplo Parameter representa a los parámetros que pueden existir en las operaciones, esto parámetros pueden ser de entrada o salida, para hacer esta distinción se crearon las clases ParameterIput y ParameterOutput ambas derivan de la clase Parmeter, de manera similar ocurre con la clase DataType. Tanto Parameter como DataType son clases abstractas para evitar que existan instancias de ellas, solo se utilizan para agrupar e introducir un orden dentro de la jerarquía de clases

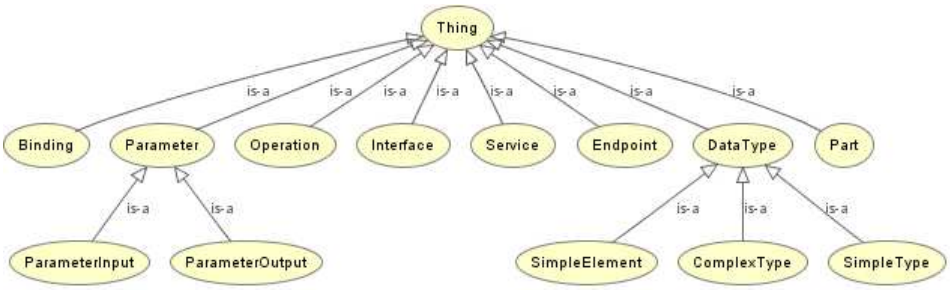

Figura 3: Modelo ontologico WSDL 1.1

Para representar las relaciones entre las clases, owl permite establecer estas relaciones utilizando propiedades de objeto (ObjectProperties) y propiedades de datos (DataProperties).

Las propiedades de objetos permiten hacer relaciones entre las clases del modelo, lo cual permite afirmar hechos generales sobre los miembros de las clases. Las propiedades de tipos de datos relacionan individuos con literales RDF o tipos simples de XML Schema. Tanto las propiedades de objetos y propiedades de datos deben tener un dominio y un rango en el cual se utilicen, en ambas un dominio de propiedad reduce los individuos a los que se puede aplicar dicha propiedad. Mientras que el rango de un ObjectProperties reduce los individuos que una propiedad puede tener como su valor, el rango de un DataProperties reduce los tipos de datos que una propiedad puede tener como su valor. La Figura 4 presenta las propiedades de objetos y las propiedades de datos que tienen las clases del modelo ontológico.

2.1.3. Extracción y representación de WSDL 2.0

En la Figura 5 se tiene la estructura general de un archivo WSDL 2.0 a partir de la cual se diseño el modelo ontológico para representar este tipo de archivos. De igual forma se modelaron las relaciones que existen entre cada sección del WSDL con los datos que contienen.

ObjectProperties	Dominio	Rango
hasBinding	Endpoint	Binding
hasDataTypeComplexType	Part	ComplexType
hasDataTypeSimpleElement	Part	SimpleElement
hasDataTypeSimpleType	Part	SimpleType
hasEndpoint	Service	Endpoint
hasInterface	Binding	Interface
hasOperation	Interface	Operation
hasParameterInput	Operation	ParameterInput
hasParameterOutput	Operation	ParameterOutput
hasPart	Parameter	Part

(a) ObjectProperties

DataProperties	Dominio	Rango
hasAttributeBinding	Binding	string
hasAttributeDataType	DataType	string
hasAttributeEndpoint	Endpoint	string
hasAttributeOperation	Operation	string
hasAttributeParameter	Parameter	string
hasAttributePart	Part	string
hasAttributeService	Service	string
hasElementComplexType	ComplexType	string
hasInterfaceName	Interface	string
hasRestrictionSimpleType	SimpleType	string

(b) DataProperties

Figura 4: ObjectProperties y DataProperties para el modelo ontologico WSDL 1.1

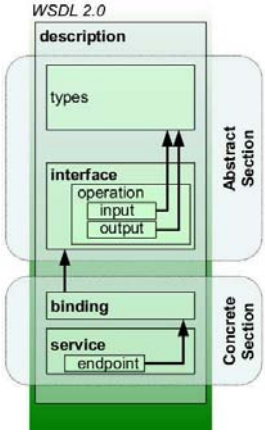

Figura 5: Estructura de un archivo WSDL 2.0

La Figura 6 muestra el diseño del modelo ontologico para representar los servicios web descritos mediante la especificación del WSDL 2.0.

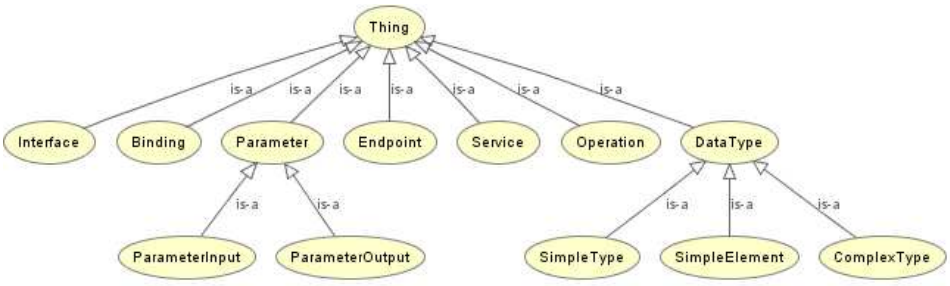

Figura 6: Modelo ontologico WSDL 2.0

La Figura 7 presenta las propiedades de objetos y las propiedades de datos que

tienen las clases del modelo ontológico WSDL 2.0.

ObjectProperties	Dominio	Rango
hasBinding	Endpoint	Binding
hasDataTypeComplexType	Parameter	ComplexType
hasDataTypeSimpleElement	Parameter	SimpleElement
hasDataTypeSimpleType	Parameter	SimpleType
hasEndpoint	Service	Endpoint
hasInterface	Binding	Interface
hasOperation	Interface	Operation
hasParameterInput	Operation	ParameterInput
hasParameterOutput	Operation	ParameterOutput

(a) ObjectProperties

DataProperties	Dominio	Rango
hasAttributeBinding	Binding	string
hasAttributeDataType	DataType	string
hasAttributeEndpoint	Endpoint	string
hasAttributeOperation	Operation	string
hasAttributeParameter	Parameter	string
hasAttributeService	Service	string
hasAttributeService	ComplexType	string
hasElementComplexType	Interface	string
hasInterfaceName	SimpleType	string

(b) DataProperties

Figura 7: ObjectProperties y DataProperties para el modelo ontologico WSDL 2.0

2.1.4. Extracción y representación de OWLS

La Figura 8 muestra la ontología principal de OWLS para representar un servicio Web. OWLS se basa en la definición de varias ontologías escritas en OWL que permiten la descripción de servicios web semánticos en diferentes niveles de abstracción.

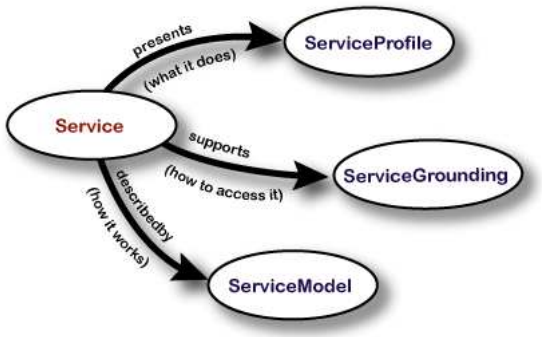

Figura 8: Ontologia Principal de OWLS para representar un servicio

La Figura 9 muestra el diseño de la ontología para representar la estructura de un archivo OWLS que describe a un servicio Web.

La Figura 10 presenta las propiedades de objetos y las propiedades de datos que tienen las clases del modelo ontológico OWLS.

2.2. Implementación

El proyecto se desarrollo en una pc con las siguientes características:

- Computadora con procesador Intel Pentium DualCore E5400, 2 GB de memoria RAM, disco duro de 320 GB.

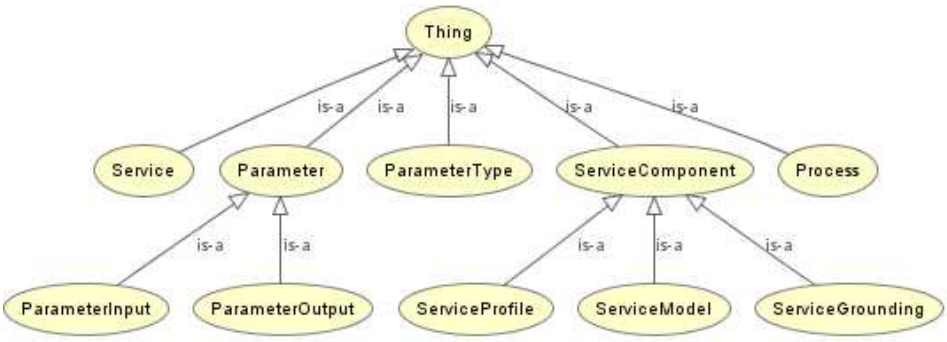

Figura 9: Modelo ontologico OWLS

ObjectProperties	Dominio	Rango
hasDescribeBy	Service	ServiceModel
hasParameterInput	Process	ParameterInput
hasParameterOutput	Process	ParameterOutput
hasParameterType	Parameter	ParameterType
hasPresents	Service	ServiceProfile
hasProcess	ServiceModel	Process
hasOperation	Service	ServiceGrounding

(a) ObjectProperties

DataProperties	Dominio	Rango
hasAttributeGrounding	ServiceGrounding	string
hasAttributeModel	ServiceModel	string
hasAttributeParameter	Parameter	string
hasAttributeProfile	ServiceProfile	string

(b) DataProperties

Figura 10: ObjectProperties y DataProperties para el modelo ontologico WSDL OWLS

- Con sistemas operativos Linux Debian 6.0 y Windows 7 professional.
- Servidor de aplicaciones Apache-Tomcat-7.0.30

2.2.1. Herramientas utilizadas

NetBeans: El proyecto se implemento utilizando java como lenguaje de programación, mediante el uso de NetBeans [8] como entorno de desarrollo integrado (IDE). El uso de este IDE permite al programador un desarrollo más ágil y una mejor estructuración del proyecto. Esta herramienta permitió una mejor documentación del código fuente de la aplicación así como la realización de las pruebas unitarias.

Mediante el uso de la opción debug se pudieron encontrar errores sobre el funcionamiento del sistema, ya que esta opción te permite marcar los puntos de interrupción en donde se tuviera algún funcionamiento inesperado del sistema. Debido a que este tipo de errores es complicado encontrar donde existe un error, con el debug se pudo seguir el cambio en las variables y ver donde se tenía un valor que no era el esperado y corregir la lógica de programación.

Protégé 4.1: Se utilizó Protégé 4.1[9] como editor de ontologías, esta herramienta

facilita la creación de los modelos ontológicos, el entorno gráfico del editor es intuitivo y tiene varias pestañas para trabajar con alguna sección específica que conforma la ontología, como puede ser las clases que conforman el modelo, las propiedades de las clases y las propiedades de tipos de datos, entre otros.

Permite agregar o eliminar clases del modelo de una forma intuitiva, también permite la creación de las propiedades de objetos y propiedades de datos, establecer las relaciones que existen entre las clases, agregar que propiedades tendrá cada objeto y definir el rango y el dominio que existe en cada relación que se defina entre las clases.

El uso de Protégé fue una parte fundamental en el desarrollo del proyecto, permitió el diseño de los modelos ontológicos, el desarrollo de los modelos fue de manera iterativa por lo que conforme se avanzaba en el proyecto se hicieron cambios a los modelos de tal manera que el diseño fuera mejorando paulatinamente conforme se encontraban mas requerimientos que agregar a los modelos.

2.2.2. APIs utilizadas

JDOM 2.0.2: Es un API implementado en java para trabajar con archivos xml permite crear, leer y manipular este tipo de documentos. Proporciona una capa de abstracción entre un parser de XML y nuestra aplicación, los parsers que se pueden utilizar con JDOM [10] son SAX y DOM. JDOM está enfocado para trabajar en java, por lo que su uso se facilita si se tiene conocimiento de este lenguaje de programación, como los archivos WSDL y OWLS están basados en xml la manipulación de estos se facilita al utilizar esta API, al leer algún documento JDOM nos proporciona clases y métodos para manipularlo, a partir de la estructura del archivo se crea un árbol basado en nodos que nos permite recorrer los nodos y recuperar la información que se necesite. JDOM ayudo en la recuperación de la información de los archivos de descripción permitiendo una navegación a través del los nodos de una forma más fácil que si se hubiera utilizado solamente un parser XML como DOM o SAX. El uso de esta API fue de vital importancia en la implementación del proyecto, debido a que una parte fundamental del proyecto es la recuperación de la información.

OWL API: Esta API proporciona clases y métodos para el trabajo con ontologías, permitiendo la creación, manipulación y lectura. Particularmente esta API es muy importante en el proyecto ya que nos permite realizar el poblado de la ontología, después de haber cargado el template del modelo ontológico y de haber obtenido la información de los archivos de descripción de servicios.

Esta API esta implementada en java lo que permitió una fácil integración con el proyecto y con JDOM, de tal manera que la integración de ambas resulto

sencilla y no se produjo ningún conflicto al trabajar con ella. OWL API [11] cumplió con el objetivo de crear una colección de servicios web con los datos obtenidos de los archivos de descripción, permitiendo establecer las relaciones definidas en el modelo ontológico.

Protege-OWL Code Generator: Esta API permite la creación de clases java que representan a una ontología desarrollada en Protégé, una vez que se tiene elaborado el modelo ontológico y habiendo definido las clases, los dataProperties, los objectProperties, etc. esta API nos permite generar las clases necesarias para posteriormente trabajar con la ontología.

Cada clase definida en el modelo se mapea a una clase concreta en java con los métodos necesarios para poder manipular los objectProperties y dataProperties especificos a esa clase.

Para la instanciación de las clases que representan a la ontología se implementa el patrón Factory en la creación de los objetos, de tal manera que cuando se necesite una instancia de un objeto determinado la clase Factory correspondiente es la encargada de retornar el objeto apropiado que se le solicite, es decir, se delega la responsabilidad de creación de los objetos al Factory específico al modelo ontológico correspondiente.

PrimeFaces 3.4: PrimeFaces [12] es una implementación de JSF 2.0 para el desarrollo de interfaces de usuario en un entorno web. Se utilizó este framework para desarrollar la página del proyecto y así poder realizar pruebas del sistema desde la Web de tal manera que pueda ser accedida por distintos usuarios interesados en el tema.

2.3. Pruebas del sistema

2.3.1. Pruebas WSDL 1.1

1. Permitir subir al servidor únicamente archivos con extensión wsdl
2. Extracción de la información y poblado de la ontología.
3. Permitir la descarga del archivo generado, es decir, la ontología poblada.
4. Eliminación de los archivos subidos al servidor.
5. Lectura del archivo generado usando Protégé 4.1
6. Mostrar en Protégé 4.1 los datos creados a partir de la recuperación de la información de los archivos de descripción wsdl.

A continuación se muestra las capturas de la prueba realizada. La Figura 11 nos muestra la interfaz inicial del sistema.

Figura 11: Estado inicial del sistema

En la Figura 12 se observan los distintos lenguajes de descripción de servicios (SDL) con los que puede trabajar el sistema. Podemos seleccionar cualquiera de las tres opciones.

Para esta prueba se utilizó la primera opción que corresponde a WSDL 1.X.

Figura 12: SDLs disponibles

Una vez seleccionado la opción WSDL 1.X, hacemos clic en el botón iniciar, esto nos mostrara el componente fileUploader para poder cargar los archivos de descripción de servicios, como lo muestra la Figura 13, para este caso nos permitirá cargar archivos WSDL.

El sistema nos muestra una advertencia por si se subirán más de 200 archivos al servidor, para que carga de los archivos se realice en bloques de no más de 200, esto con el fin de evitar la carga del explorador que se esté usando.

Para buscar los archivos que se van a utilizar presionamos el botón buscar, esta acción nos abrirá el explorador de archivos de nuestro sistema operativo, de tal manera que podamos navegar hasta la ruta donde se encuentre la colección, posteriormente seleccionamos el o los archivos que se subirán al sistema y damos clic en el botón abrir, así como lo muestra la Figura 14.

Una vez que presionamos el botón abrir se nos mostraran en el componente los archivos seleccionados. Y procedemos a subir los archivos al servidor, esto se realiza al presionar el botón subir.

En la Figura 15 se puede observar los errores que se producen cuando queremos subir al servidor archivos con una extensión diferente a WSDL.

Figura 13: Componente para subir los archivos

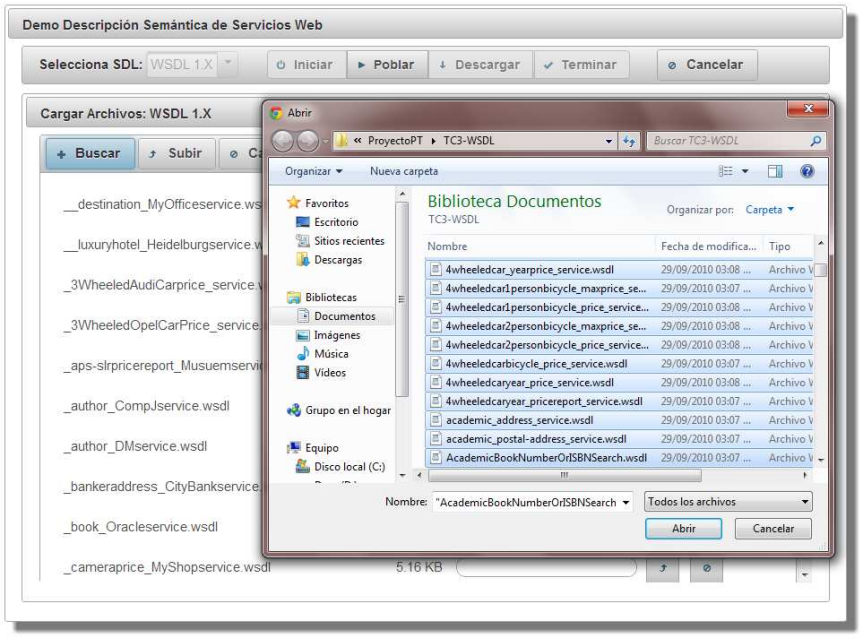

Figura 14: Selección de archivos WSDL

La Figura 16 nos muestra una lista con los nombres de los archivos que se subieron al servidor y el total de ellos. Si se requiere subir más archivos volvemos a realizar los pasos anteriores, es decir, buscar, seleccionar y subir los archivos.

Una vez realizado la carga de archivos procedemos a poblar la ontología, para empezar a realizar este proceso damos clic en el botón poblar. El proceso de poblado implica la extracción de la información de cada uno de los archivos, la inserción de esta información en el modelo ontológico y guardar los cambios que se realizan al modelo.

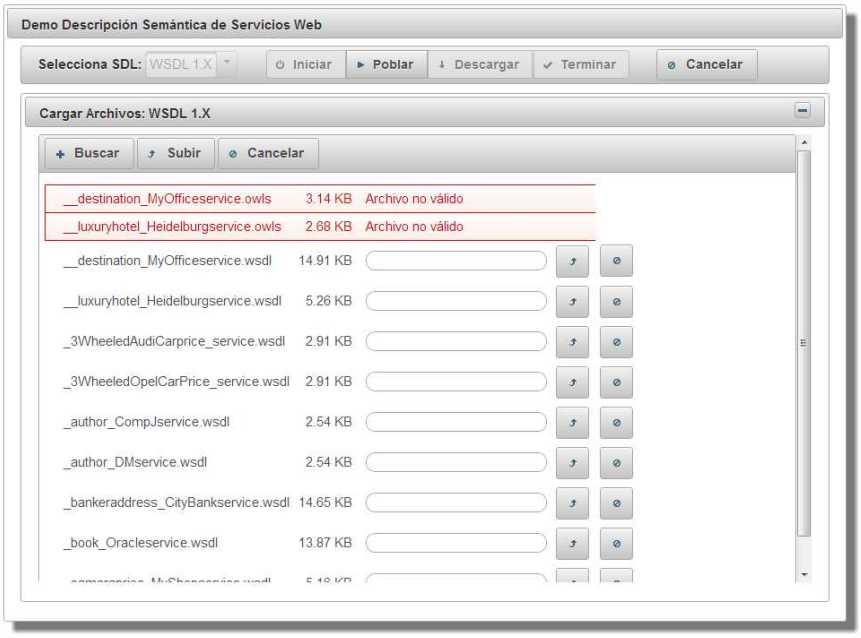

Figura 15: Errores relacionados con la extensión de archivos

En la Figura 17 se observa el estado de la interfaz después de hacer clic en el botón poblar, se nos muestra una ventana modal que nos indica que el sistema está trabajando, el tiempo que tarda la interfaz en terminar el proceso, depende del número de archivos con los que se trabaje.

La Figura 18 nos muestra la interfaz después de haber terminado el poblado sin que se haya producido algún error durante el procesamiento, se puede observar que se han habilitado los botones Descargar y Terminar, el botón Descargar nos permite guardar en nuestra computadora el archivo OWL que contiene toda la información que se recupero de los archivos de descripción. Finalmente damos clic en el botón Terminar para volver al estado inicial de la interfaz y poder trabajar con otro SDL.

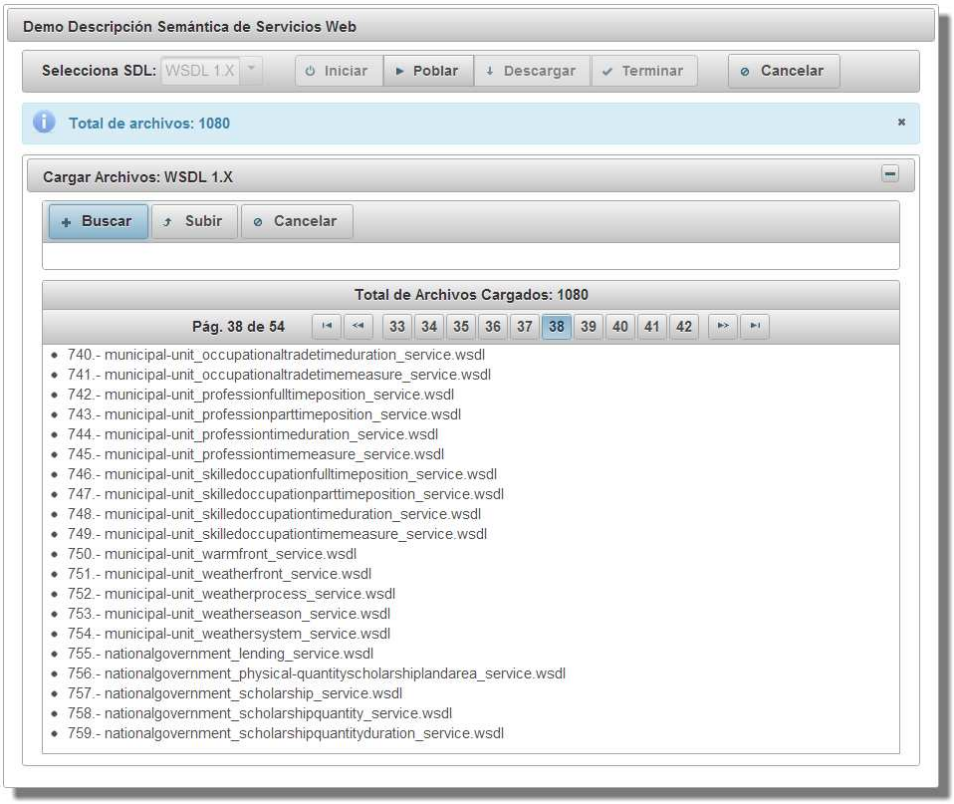

Figura 16: Total de archivos cargados

Si durante el poblado se produjo algún error los archivos que produjeron el error no son procesados, son ignorados y el proceso continua con el siguiente fichero que sea válido y seguir con el poblado. En la Figura 19 se observa la interfaz del sistema cuando se produjeron errores, el sistema nos informa los nombres de los archivos en los que se produjeron los errores.

Después de haber terminado el proceso de poblado, ya sea con errores o sin errores, descargamos el archivo generado por el sistema. En la Figura 20 se observa el archivo descargado.

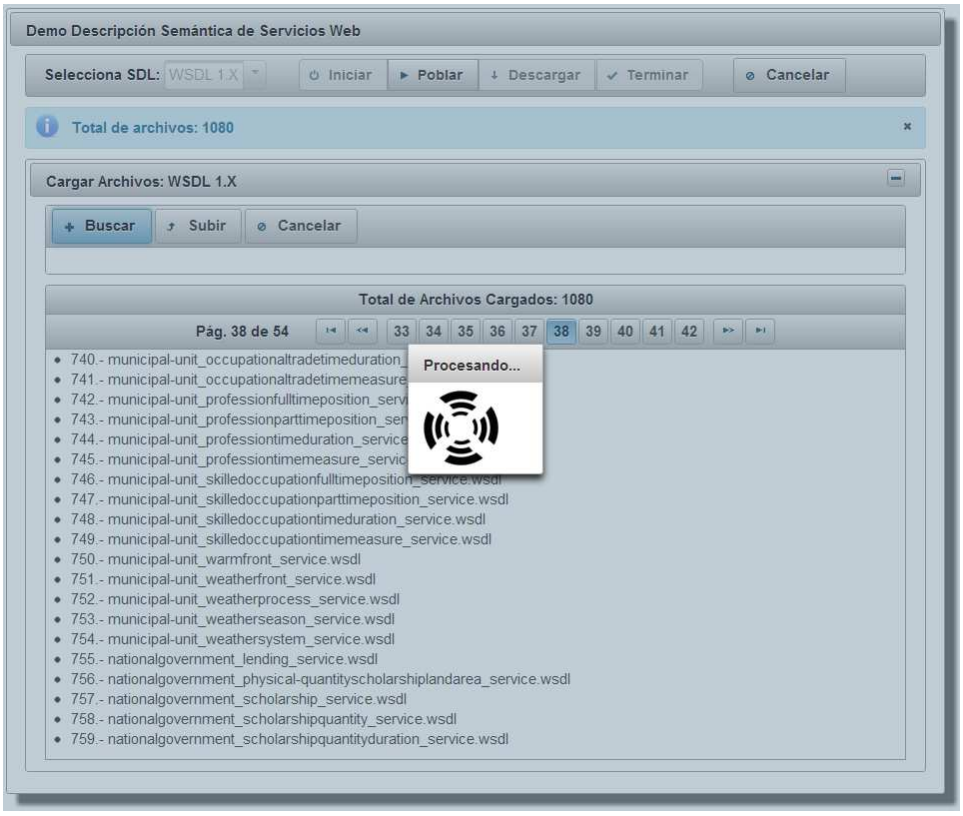

Figura 17: Estado de la interfaz mientras se realiza el poblado

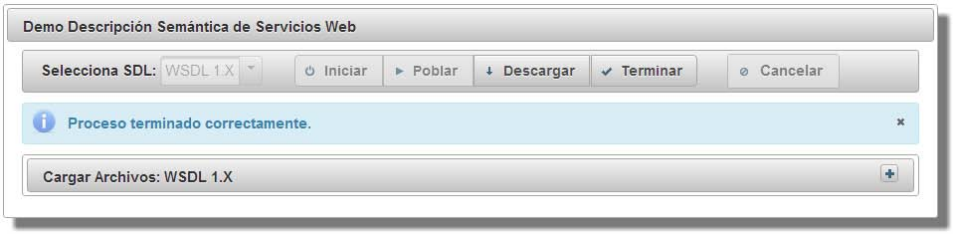

Figura 18: Estado de la interfaz al terminar el poblado correctamente

Después de haber realizado la descarga de la ontología procedemos a abrir el archivo, Protégé 4.1 nos permite trabajar con ontologías, por lo que utilizamos este programa para abrir el archivo OWL. La Figura 21 muestra la interfaz inicial de Protégé , seleccionamos la opción open OWL ontology, y procedemos a buscar el archivo que hemos descargado anteriormente y damos clic en abrir.

La Figura 22 nos muestra la vista Entities de la interfaz de Protégé después de haber abierto el archivo, podemos observar las clases que conforman la ontología

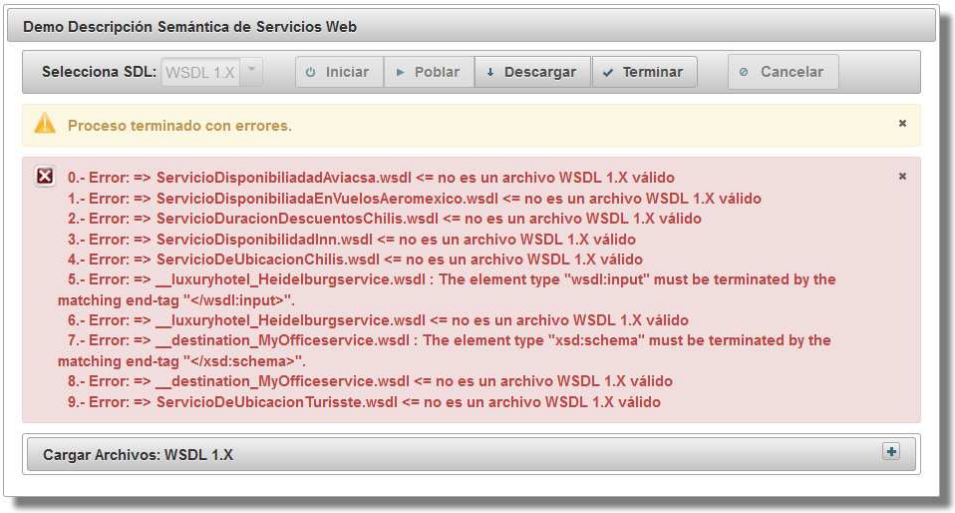

Figura 19: Estado de la interfaz al terminar el poblado con errores

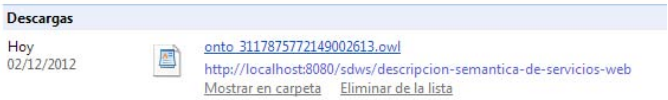

Figura 20: Archivo OWL descargado

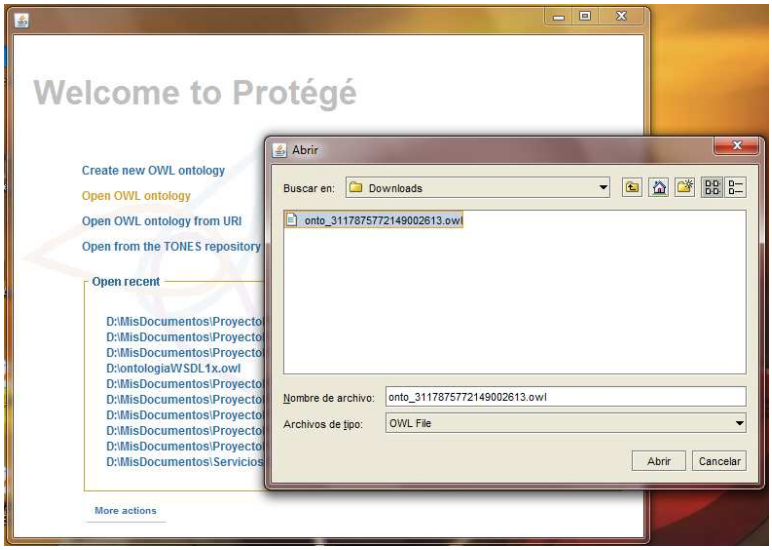

Figura 21: Abrir ontologia con Protégé

así como los individuos que existen por cada clase.

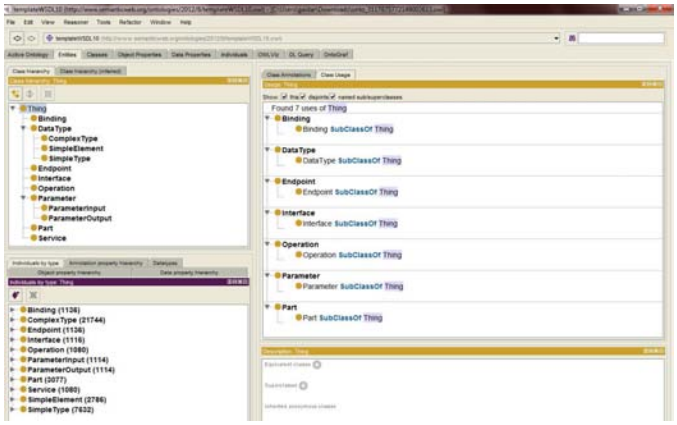

Figura 22: Vista Entities con Protégé

En la Figura 23 se observa la vista Individuals de Protége, esta vista nos muestra el nombre de los individuos de la clase que se haya seleccionado, es decir, nos muestra los miembros que pertenecen a la clase seleccionada. También nos proporciona toda la información asociada a un individuo que se seleccione.

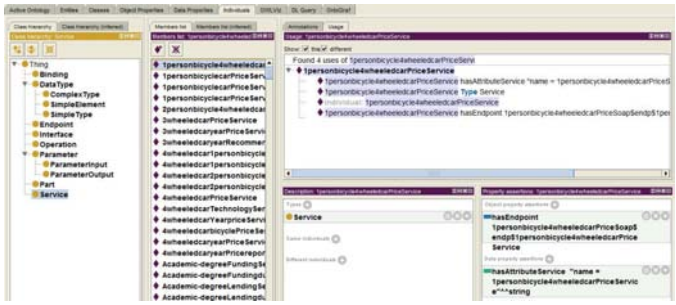

Figura 23: Vista individuals con Protégé

En la Figura 24 se muestra la vista OntoGraf que nos proporciona Protége, en esta imagen se aprecia de forma grafica las relaciones que existen entre las clases y los individuos que conforman la ontología, del lado derecho se muestra las relaciones que representa cada una de las flechas dentro del grafo.

2.3.2. Pruebas WSDL 2.0

1. Permitir subir al servidor únicamente archivos con extensión wsdl
2. Extracción de la información y poblado de la ontología.
3. Permitir la descarga del archivo generado, es decir, la ontología poblada.

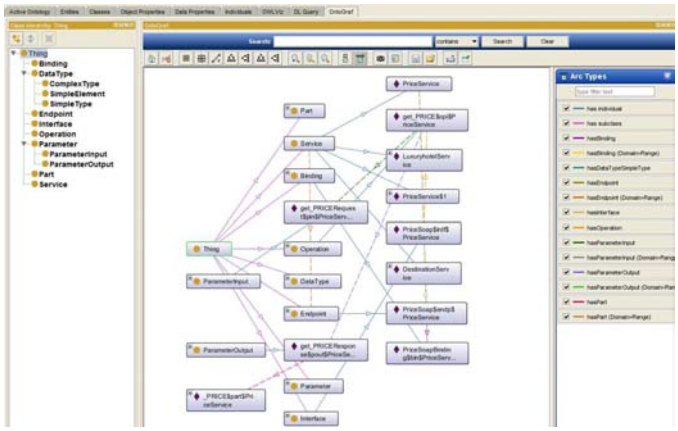

Figura 24: Vista OntoGraf con Protégé

4. Eliminación de los archivos subidos al servidor.
5. Lectura del archivo generado usando Protégé 4.1
6. Mostrar en Protégé 4.1 los datos creados a partir de la recuperación de la información de los archivos de descripción wsdl

La Figura 25 se muestra la interfaz al seleccionar la opción WSDL 2.0

Figura 25: Selección de SDL WSDL 1.1

La Figura 26 muestra el estado de la interfaz después de dar clic en el botón iniciar, se habilita el componente fileUploader para cargar los archivos.

La Figura 27 muestra la carga de nuestra colección WSDL 2.0 al servidor.

La Figura 28 muestra el total de archivos que se subieron al servidor.

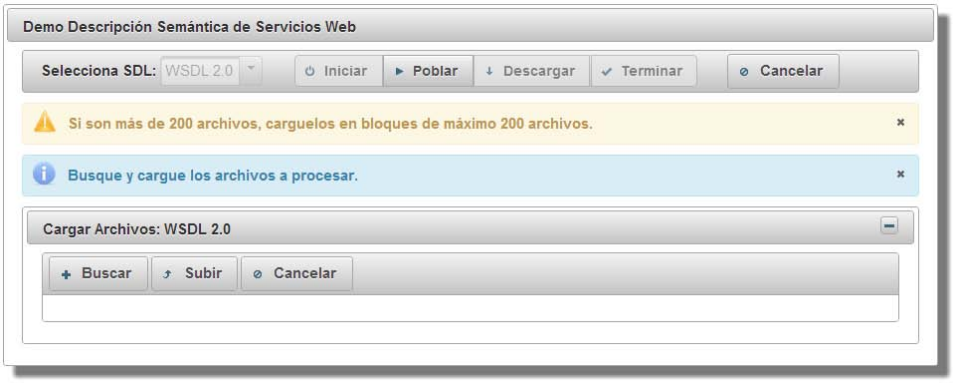

Figura 26: Componente para subir los archivos

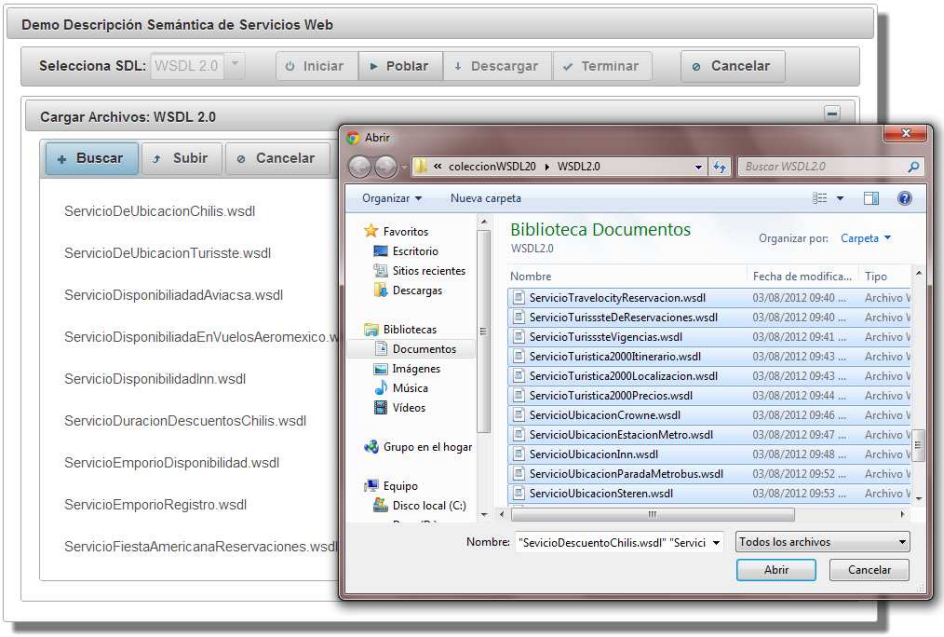

Figura 27: Selección de archivos

Después de terminar la carga de archivos damos clic al botón poblar y esperamos a que termine el proceso de poblado, al terminar el estado de la interfaz es la que se observa en la Figura 29, se habilitan los botones de Descargar y Terminar.

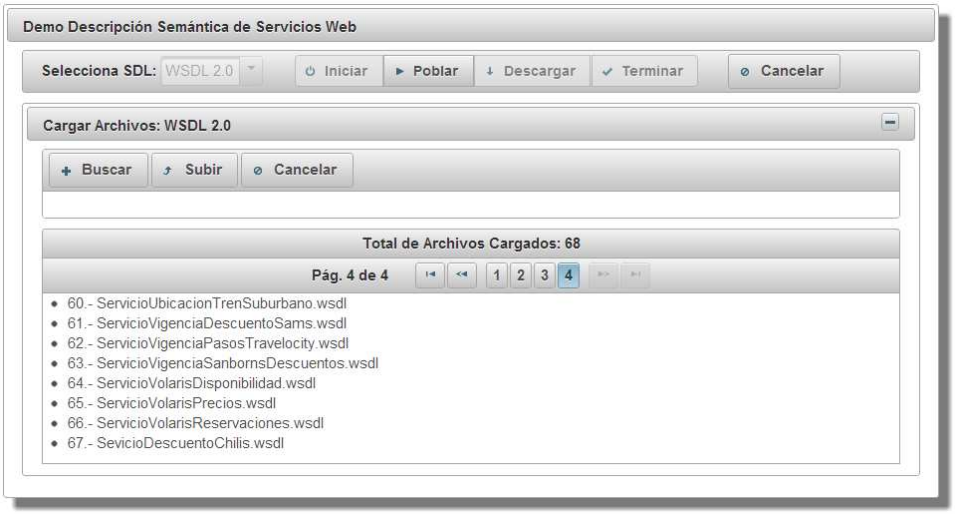

Figura 28: Total de archivos cargados

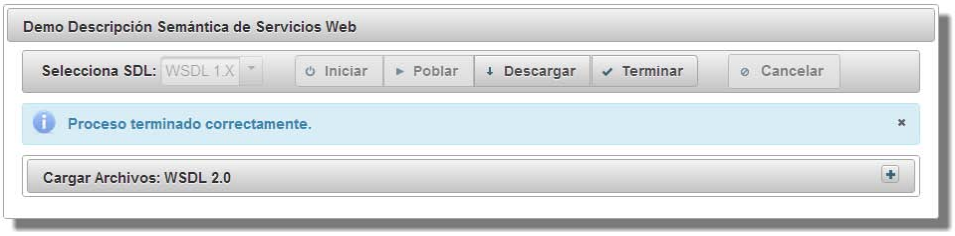

Figura 29: Estado de la interfaz al terminar el poblado correctamente

2.3.3. Pruebas OWLS

1. Permitir subir al servidor únicamente archivos con extensión owls
2. Extracción de la información y poblado de la ontología.
3. Permitir la descarga del archivo generado, es decir, la ontología poblada.
4. Eliminación de los archivos subidos al servidor.
5. Lectura del archivo generado usando Protégé 4.1
6. Mostrar en Protégé 4.1 los datos creados a partir de la recuperación de la información de los archivos de descripción wsdl.

La Figura 30 muestra la interfaz al seleccionar la opción OWLS

Figura 30: Selección de SDL OWLS

La Figura 31 muestra el estado de la interfaz después de dar clic en el botón iniciar, se habilita el componente fileUploader para cargar los archivos.

Figura 31: Componente para subir los archivos

La Figura 32 muestra la carga de nuestra colección OWLS al servidor.

La Figura 33 muestra el total de archivos que se subieron al servidor.

Después de terminar la carga de archivos damos clic al botón poblar y esperamos a que termine el proceso de poblado, al terminar el estado de la interfaz es la que se observa en la Figura 34, se habilitan los botono de Descargar y Terminar.

Descargamos el archivo que se generó, la Figura 35 nos muestra el archivo descargado.

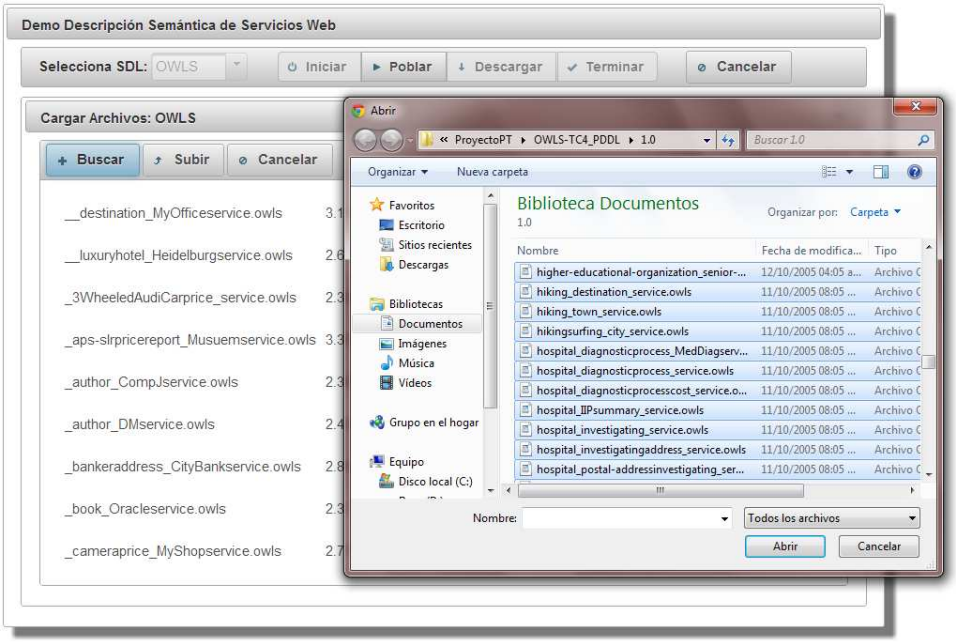

Figura 32: Selección de archivos OWLS

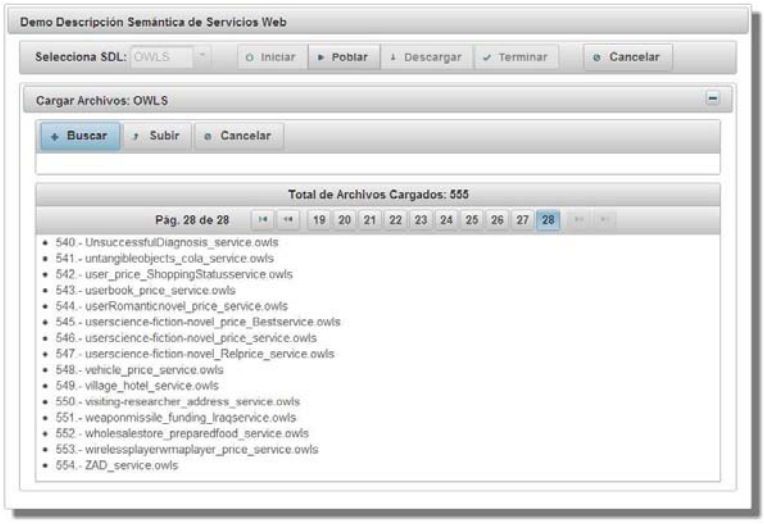

Figura 33: Total de archivos cargados

Abrimos el archivo descargado anteriormente usando Protégé, seleccionamos la opción Open OWL ontology y buscamos la ubicación del fichero tal como se muestra en la Figura 36.

La Figura 37 muestra el contenido del archivo OWL que descargamos.

Figura 34: Estado de la interfaz al terminar el proceso correctamente

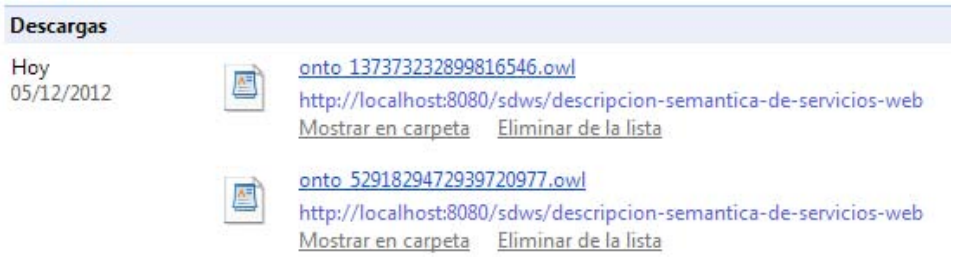

Figura 35: Archivo OWLS descargado

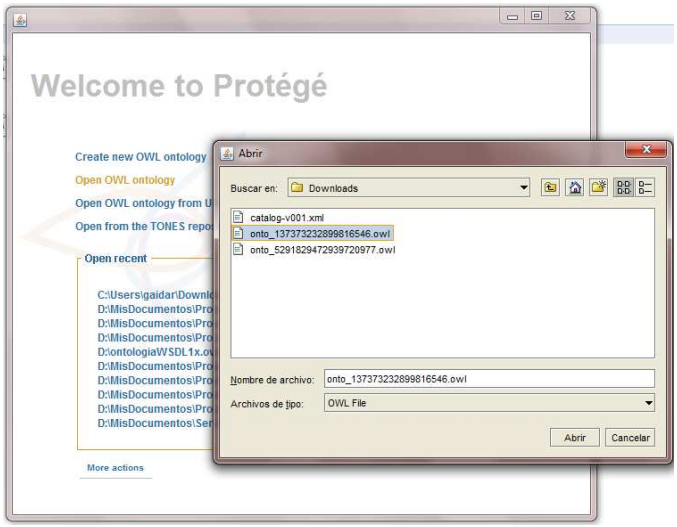

Figura 36: Abrir ontologia con Protégé

3. Conclusiones

Al finalizar este proyecto terminal se obtuvieron conocimientos más concretos acerca de lo que son las ontologías, como diseñarlas, como trabajar con ellas. Nos

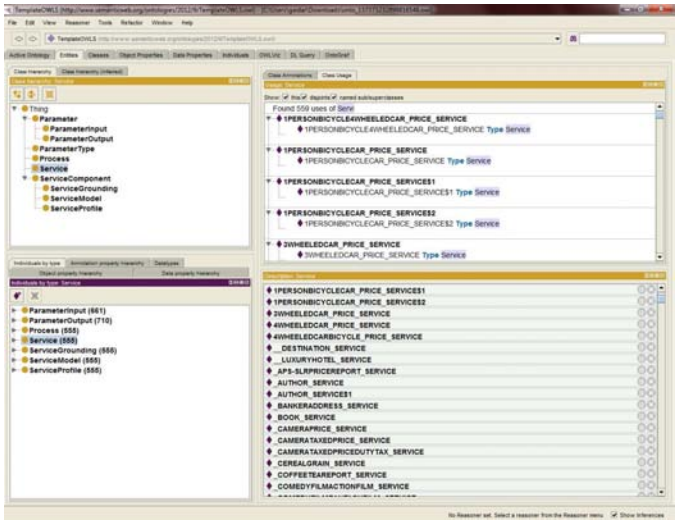

Figura 37: Vista Entities con Protégé

dimos cuenta el papel que juegan dentro de la web semántica, de cómo a través de estas se puede dotar de contenido semántico al área del conocimiento que se modele.

Aprendí que para poder diseñar una ontología primero se debe de tener bien delimitado el área del conocimiento que se quiere modelar, además como el diseño puede ir cambiando conforme se encuentren otros requerimientos que se necesiten representar, el modelado se convierte en un proceso iterativo.

Una dificultad que se encontró en el desarrollo del proyecto fue la recuperación de la información de los archivos de descripción de los servicios, si bien existen APIs especializadas para trabajar con archivos WSDL, estas agregaban una mayor complejidad al proyecto.

Por ejemplo el API EasyWSDL permite extraer información del archivo de descripción, para realizar esto se lee el WSDL y se crean clases que representan la estructura de un archivo WSDL, sin embargo, la sección Types ocupa un tratamiento especial debido a la complejidad de los datos que se pueden definir en el, por lo que se tenía que volver a cargar el archivo WSDL utilizando la API EasySchema para recuperar esos datos, esto represento un problema en el rendimiento de la aplicación al cargar en memoria dos veces el mismo archivo. Además si se utilizaba esta API solo nos serviría para recuperar información de los WSDL y cuando se tratara de un archivo OWLS no nos serviría de nada.

Debido a que el proyecto solo requiere la recuperación de los datos, es decir, solo hacemos lectura de los archivos sin modificarlos, se opto por trabajar con JDOM debido a que nos permite recorrer la estructura de los archivos de manera independiente de si se trata de un WSDL o de cualquier otro archivo basado en xml, por lo que se selecciono esta API debido a que el proyecto también trabaja

con archivos OWLS, JDOM nos permitió trabajar con ambos tipos de archivos, si bien los WSDL 1.X, WSDL 2.0 y los OWLS tienen estructuras distintas, estas están basados en XML por lo que JDOM puede crear un árbol DOM que represente a cada tipo de archivo y nos permite utilizar las clases y métodos para poder recuperar la información que se necesite.

Con JDOM al recuperar los datos de la sección Types del WSDL, nos permitió recorrerlo como cualquier nodo, sin la necesidad de crear clases específicas para representar los tipos de datos. Además como solo se requiere extraer información de los archivos, sin la necesidad de modificarlos JDOM resulto ser más eficiente al realizar el parseo de los archivos de descripción.

Otra de las dificultades importantes que se encontró en el desarrollo del proyecto, fue como realizar el poblado de las ontologías, es decir, una vez diseñado el modelo ontológico y creado el parser para recuperar los datos de los archivos de descripción, como hacer posible que los datos extraídos se guardaran en el modelo ontológico.

Esta problemática se solvento utilizando Protégé OWL API esta API nos permite poder cargar el archivo que representa el modelo ontológico y trabajar con él, nos permite editarlo, modificarlo, etc. Como cada modelo es diferente se implemento una clase por cada modelo para realizar el poblado de la ontología correspondiente.

El producto obtenido con la finalización del proyecto puede tener varias aplicaciones entre ellas y para la cual fue pensada es que pueda ser usado para crear repositorios de servicios web descritos semánticamente, a partir de archivos de descripción de servicios Web escritos en los lenguajes de descripción de servicios como son WSDL 1.X, WSDL 2.0 y OWLS, y se crearán de manera automatizada proporcionar los archivos de origen.

Posteriormente esto permitirá realizar búsquedas sobre la colección generada, las búsquedas podrán realizarse mediante el uso de reglas de inferencia sobre la ontología, además debido a la naturaleza de las ontologías y a que estas permiten la importación de otras, los modelos diseñados pueden ser importados para crear un solo modelo a partir de los que se diseñaron, logrando con esto crear una colección a partir de diferentes tipos de archivos de descripción de servicios.

Al poder realizar inferencia sobre las ontologías diseñadas, otra aplicación que se le puede dar al proyecto es poder buscar servicios Web que cumplan con ciertos requerimientos, de forma semiautomática y poder ayudar en la composición de servicios, esto mediante el uso de reglas de inferencia sobre la ontología.

Anexos

A. Reglas de nombrado de individuos

A.1. Modelo ontológico WSDL 1.1

Para nombrar a los individuos se utilizó la siguiente nomenclatura, si se intenta crear un individuo de tipo Service y dentro de la ontología ya existe uno con el mismo nombre, se le agrega un numero consecutivo al nombre del individuo, de esta forma garantizamos que existan individuos únicos y evitamos conflictos con los nombres. Identificadores de Recursos Internacionales (IRIs). IRI es el identificador de la ontología correspondiente.

Nombrado de los servicios.

IRI + nombre del servicio + $numero consecutivo

Nombrado de los demás individuos.

IRI + nombre del individuo + $ prefijo $ + nombre del servicio

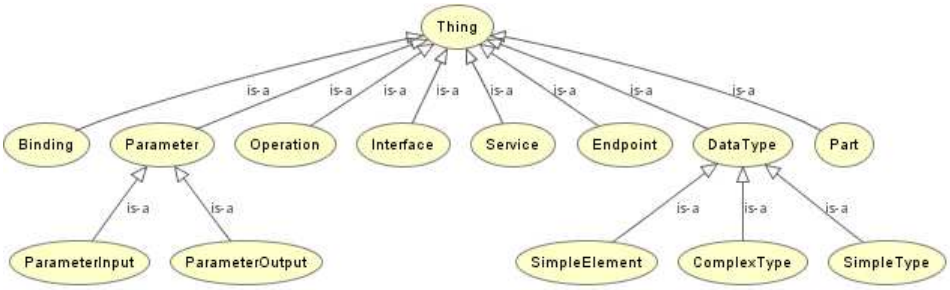

Figura 38: Modelo Ontológico WSDL 1.1

Clase	Prefijo	Clase	Prefijo
Binding	bin	Operation	opi
Interface	intf	Services	
Endpoint	endp	Part	part
ParameterInput	pin	ParameterOutput	pout
SimpleElement	se	ComplexType	ct
SimpleType	st		

Cuadro 3: Clases y prefijos WSDL 1.1

A.2. Modelo ontológico WSDL 2.0

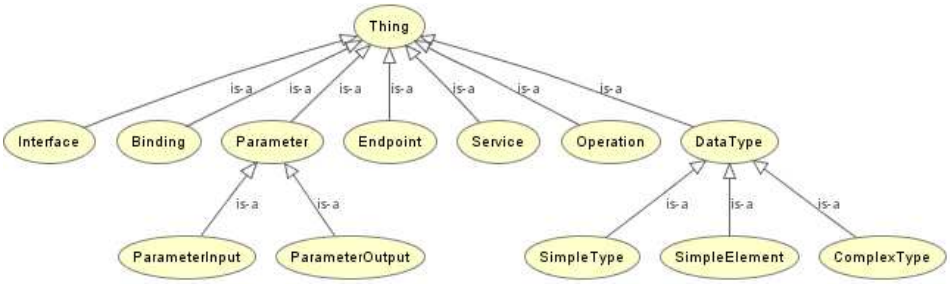

Figura 39: Modelo Ontologico WSDL 2.0

Clase	Prefijo	Clase	Prefijo
Interface	intf	Binding	bin
Endpoint	endp	Service	
Operation	opi	ParameterInput	pin
ParameterOutput	pout	SimpleType	st
SimpleElement	se	ComplexType	ct

Cuadro 4: Clases y prefijos WSDL 2.0

A.3. Modelo ontológico OWLS

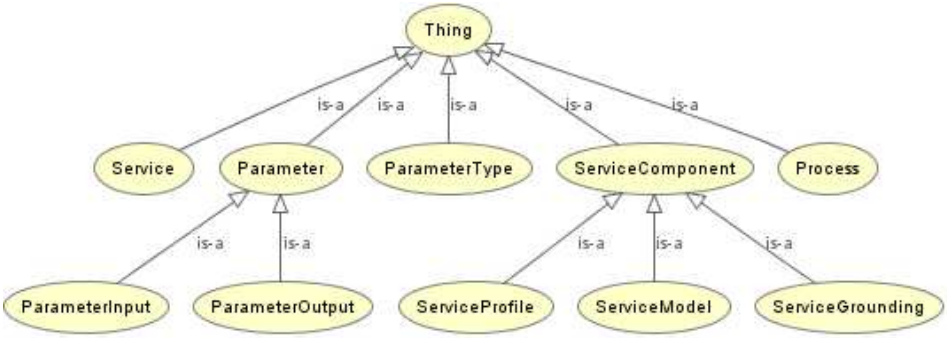

Figura 40: Modelo Ontologico OWLS

Clase	Prefijo	Clase	Prefijo
Service		ParameterType	
Process	process	ParameterInput	pin
ParameterOutput	pout	ServiceProfile	sp
ServiceModel	sm	ServiceGrounding	sg

Cuadro 5: Clases y prefijos OWLS

B. Diagramas de clases de las ontologías

B.1. Diagrama de clases WSDL 1.1

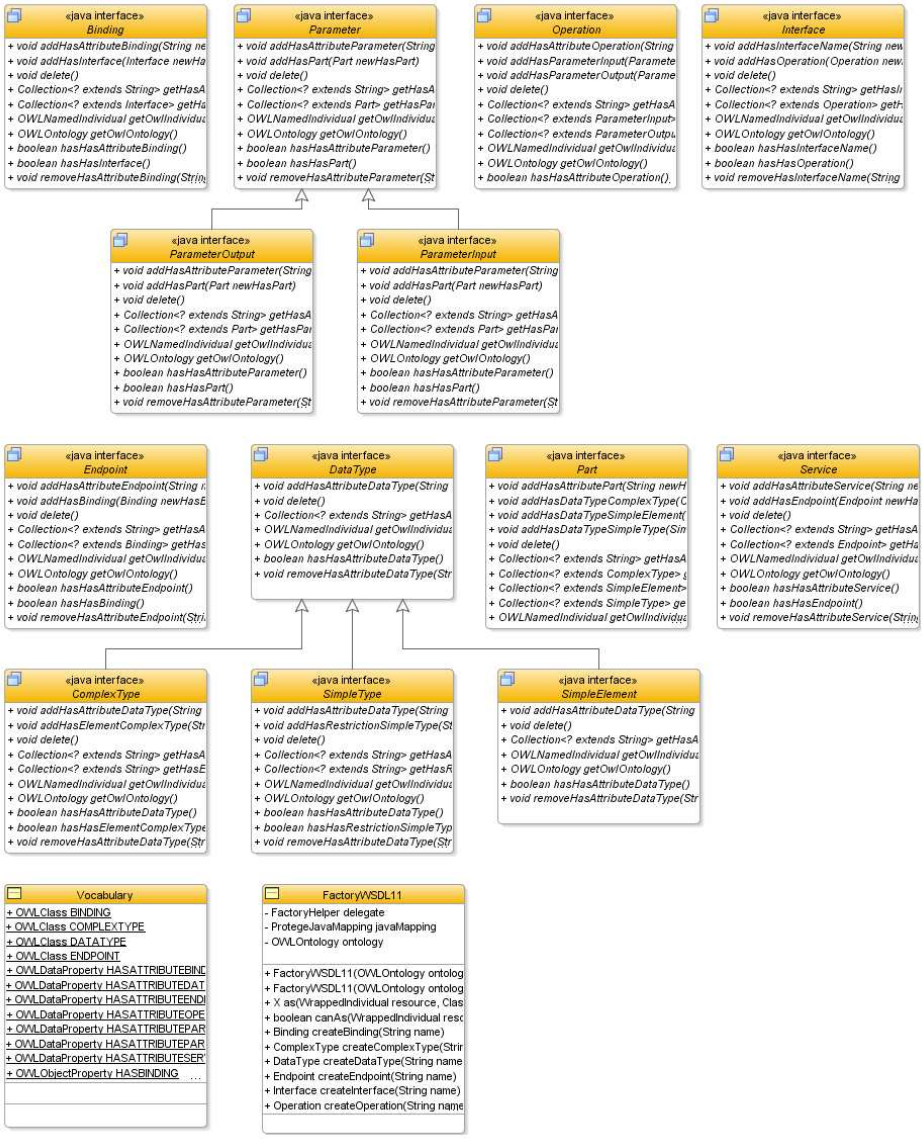

Figura 41: Diagrama de clases para el modelo ontologico WSDL 1.1

B.2. Diagrama de clases WSDL 2.0

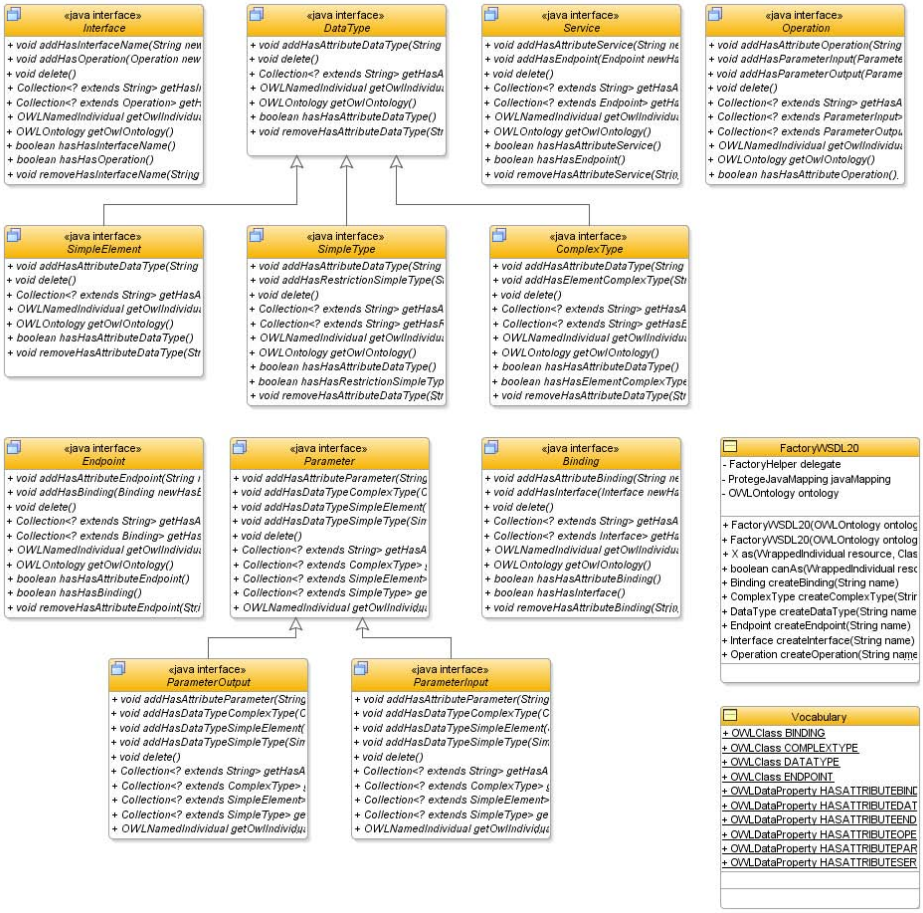

Figura 42: Diagrama de clases para el modelo ontologico WSDL 2.0

B.3. Diagrama de clases OWLS

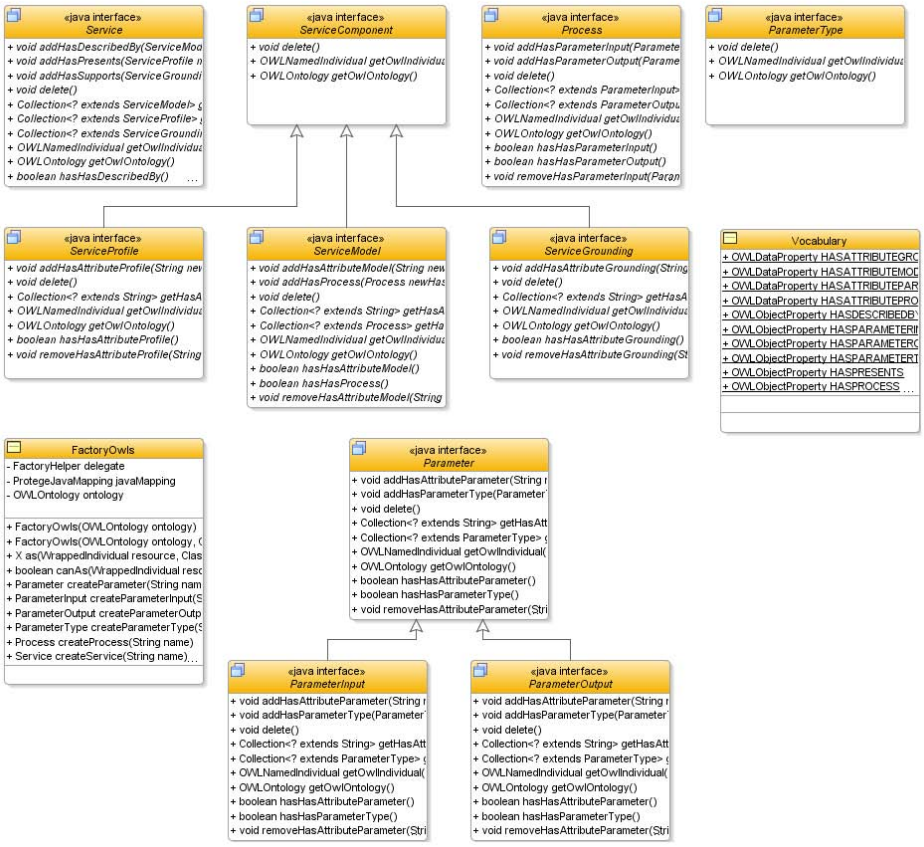

Figura 43: Diagrama de clases para el modelo ontologico OWLS

Referencias

[1] C. Lee B. Jeong, H. Cho. On the functional quality of service (fqos) to discover and compose interoperable web services. *International Journal of Expert Systems with Applications*, 2009.

[2] F. Van Harmelen D. McGuinnes. Owl web ontology language [En linea]. "Website", Noviembre 2012. `http://www.w3.org/TR/owl-features/`.

[3] E. Christensen. Web services description language (wsdl) 1.0 [En linea]. "Website", Noviembre 2012. `http://xml.coverpages.org/wsdl20000929.html`.

[4] D. Booth. Web services description language (wsdl) version 2.0 [En linea]. "Website", Noviembre 2012. http://www.w3.org/TR/wsdl20-primer/.

[5] D. Martin. Owl-s semantic markup for web services [En linea]. "Website", Noviembre 2012. http://www.w3.org/Submission/OWL-S/.

[6] M.M Ramírez. *"Plataforma de archivos de acceso remoto mediante servicios Web*. PhD thesis, Universidad Autónoma Metropolitana Azcapotzalco, D.F., México, 2008.

[7] R. Lara et al. A conceptual comparison between wsmo and owl-s [En linea]. "Website", Noviembre 2012. http://www.wsmo.org/2004/d4/d4.1/v0.1/20050106/d4.1v0.1_20050106.pdf.

[8] NetBeans.org. Primefaces[En linea]. "Website", Noviembre 2012. http://netbeans.org/.

[9] Stanford Center for Biomedical Informatics Research. Primefaces[En linea]. "Website", Noviembre 2012. http://protege.stanford.edu/.

[10] JDOM. Jdom [En linea]. "Website", Noviembre 2012. http://www.jdom.org/.

[11] OWL api. Owl api[En linea]. "Website", Noviembre 2012. http://owlapi.sourceforge.net/.

[12] Primefaces.org. Primefaces[En linea]. "Website", Noviembre 2012. http://primefaces.org/.