UNIVERSIDAD CENTRAL "MARTA ABREU" DE LAS VILLAS

Facultad de Matemática, Física y Computación

INGENIERÍA INFORMÁTICA

Trabajo de Diploma

Incorporación de semántica a los procesos de gestación y aprobación de programas de postgrado en la UCLV.

Autora: Claudia Rodríguez Ponce

Tutores: M.Sc. Carlos Alberto Pereira Marín

Dr.C. Rosendo Moreno Rodríguez

Santa Clara

Curso 2015-2016

Dedicatoria

Mi madre por todo su apoyo, amor y confianza.

Mi padre por no importar la distancia para hacerme saber cuán importante soy para él, por su apoyo y amor.

Agradecimientos

Han sido muchas las personas que de una manera u otra me brindaron su ayuda en la realización del presente trabajo, a todos ellos muchas gracias. En especial quiero agradecer a:

A mi familia: mi mamá y mi papá, por enseñarme a ser una persona de bien, por sus sabios consejos, por siempre confiar en mí, por su apoyo incondicional en todas las etapas, por sus palabras de alientos en los momentos difíciles.

A mi tutor M.Sc. Carlos Alberto Pereira Marín por su apoyo incondicional.

A Rachel Pairol por brindarme su ayuda en el desarrollo del Trabajo.

A todos los profesores que me aconsejaron durante el desarrollo de este Trabajo: Lic. Lisandra Díaz de la Paz, Lic. Juan Luis García Mendoza y Lic.Yaisel Nuñez Garcia.

A todos mis amigos, en especial a: Katlin, Andy, Emilio, Joey, Aurea, a todos mis compañeros de aula quienes han estado constantemente al tanto de la evolución de mi trabajo.

Resumen

En la presente contribución se aborda la problemática referente a la incorporación de semántica de una serie de servicios *web* que deben ser implementados siguiendo los principios de la arquitectura orientada a servicios. Un servicio *web* es un componente al que podemos acceder mediante protocolos de *web* y el estándar *XML*, como metalenguaje, para el intercambio de información. El diseño, la composición y la implementación de los servicios *web* son los primeros objetivos a cumplirse, para luego aplicar los modelos ontológicos referentes a los procesos de gestación de programas de postgrado académicos en la UCLV y de descripción semántica de los servicios utilizados en estos procesos de negocio. Entre las diferentes metodologías para la creación de servicios *web* se utilizó la denominada estrategia ágil, mientras que para la incorporación de la semántica se aplicó el modelo *OWL-S*. Varias herramientas fueron utilizadas en el propósito de este trabajo, ellas fueron, *Eclipse Juno*, *Axis2*, *AltovaXML*, *Protégé* y *ApacheTomcat*. La plataforma de programación utilizada fue *Java*. Un total de 7 servicios fueron implementados en la capa de aplicación, los cuales se usan y reutilizan en los 6 servicios de la capa de composición, estos últimos junto a los primeros son aplicados en los 14 servicios definidos en la capa de negocio. La incorporación de semántica de los meta-datos asociados a un servicio *web* permiten un mejor descubrimiento y consumo por un agente *software* de forma automatizada.

Abstract

In this contribution the problems of the semantic incorporation to a series of under SOA principles implemented web services was concerning. A web service is a component that can be accessed through standard web protocols by using XML as meta-language to exchange information. The design, composition and deployment of web services are the first goals to be accomplished, then apply the ontological models concerning the processes of gestation of postgraduate academic programs at UCLV and the semantic description of web services used in these business processes. The so-called agile strategy was used among different methodologies for creating web services, while for the incorporation of the OWL-S semantic model was applied. Several tools were used in the purpose of this work, they were, Eclipse Juno, Axis2, AltovaXML, Protégé and ApacheTomcat. The programming Java platform was used. A total of 7 services were implemented in the application layer, which are used and reused by the 6 services defined on the layer composition, these latter and the first mentioned ones are applied in all 14 services defined at the business layer. Incorporating semantics of the meta-data associated to a web service enable better discovery and consumption by an automated software agent.

Tabla de contenido

Lista de figuras

Introducción

La gestión de procesos de negocio o *BPM* (siglas en inglés para *Business Process Management*) es una disciplina que considera tanto el uso de las metodologías de mejora de procesos, como el uso de herramientas computacionales que soportan las fases definidas en las metodologías para la adopción del *BPM*, tomando en cuenta aspectos sociales y tecnológicos. El *BPM* tiene como objetivo ayudar a automatizar, administrar y optimizar procesos; además, intenta facilitar la interacción entre los humanos y las herramientas requeridas para operar los procesos de negocio. La aplicación de tecnologías de *BPM* ha aumentado considerablemente en los últimos años. Como consecuencia, existe una gran variedad de herramientas que apoyan en las distintas fases del ciclo de vida de los procesos de negocio. Sin embargo, muchas de estas herramientas no toman en cuenta el conocimiento organizacional como directriz en el desarrollo de los procesos (ORACLE, 2008).

La arquitectura orientada a servicios o *SOA* (siglas en inglés para *Service Oriented Architecture*) supone una estrategia general de organización de los elementos de tecnología de información o *IT* (siglas en inglés para *Information Technologies*), de forma que una colección complicada de sistemas distribuidos y aplicaciones complejas se pueda transformar en una red de recursos integrados, simplificada y sumamente flexible. Un proyecto *SOA* bien ejecutado permite alinear los recursos de *IT* de forma más directa con los objetivos de negocio, ganando así un mayor grado de integración con clientes y proveedores, además proporciona una inteligencia de negocio más precisa y más accesible, con la cual se podrán adoptar mejores decisiones y ayuda a las empresas a optimizar sus procesos internos y sus flujos de información para mejorar la productividad individual. El resultado neto es un aumento muy notable de la agilidad de la organización (MS, 2006).

Ambos paradigmas tecnológicos son de por sí disciplinas divergentes, pero la combinación *BPM-SOA* permite utilizar a los servicios como componentes reusables, que pueden ser orquestados para que sirvan de soporte de los procesos de negocio dinámicos. Esta combinación permite también el diseño iterativo y optimizar los procesos basados en servicios que pueden invocarse de forma rápida (Sarwar_Bajwa *et al.*, 2008).

Con la aparición de los servicios *web* se abre la posibilidad de conseguir la automatización de tareas entre plataformas y servicios diferentes. Para poder llevar a cabo este cometido, se requiere de una plataforma con una semántica bien definida, que permita el descubrimiento y la composición de

manera automática de los servicios. La siguiente generación *web*, proporcionará la infraestructura necesaria para lograr este objetivo mediante la utilización de los servicios *web* semánticos (denominados *SWS*) (Valledor_Pellicer, 2006).

La llamada *web* semántica define y enlaza los datos en la *web* de manera que puedan ser usados de forma más efectiva para un descubrimiento, una automatización, una integración y una reutilización entre diferentes aplicaciones. El reto es ofrecer el lenguaje que exprese tanto datos como reglas para razonar sobre los datos, y que las reglas sobre cualquier sistema de representación puedan ser exportadas a la *web*. Diferentes sistemas *web* pueden utilizar disímiles identificadores para un mismo concepto. Una solución a este problema es incluir un elemento a la *Web* Semántica, colecciones de información denominadas ontologías (Samper_ Zapater, 2005).

En el empeño de lograr sistemas de información competentes, cada vez más adaptables a los dinámicos cambios en los procesos y la tecnología, el Ministerio de Educación Superior d Cuba (MES) ha apoyado con financiación a proyectos ramales de investigación para el desarrollo software o sistemas de información. La Universidad Central "Marta Abreu" de Las Villas (UCLV), a través del Departamento de Desarrollo de Sistemas, lidera la creación de un sistema de control del postgrado sobre tecnologías de software libre con ambiente *web* (Mora_Cuellar, 2010).

Los diferentes resultados obtenidos en la creación de un nuevo sistema para la gestión de la información y el control de los procesos o actividades de postgrado han sufrido cambios, en los cuales las versiones posteriores han sido creadas casi desde cero, principalmente por la incompatibilidad de las nuevas tecnologías. También los cambios en los procedimientos, regulaciones y resoluciones de las actividades de postgrado constituyeron causas para el surgimiento de nuevos sistemas.

Entre los recientes paradigmas o modelos utilizados para la creación de los sistemas está el análisis y diseño basado en *UML* (Mora_Cuellar, 2010), de forma paralela se ha trabajado en la organización del proceso de gestión de toda la información de esta actividad (Benítez_Concepción, 2010, Fernández, 2013, Mora_Cuellar, 2010), así como también en las definiciones de una ontología de dominio (Falcón_Espinosa, 2009) y de una arquitectura orientada a servicios (Mora_Cuellar, 2010).

Una arista a enfrentar en el diseño de aplicaciones o sitios *web* semánticos desde el enfoque *BPM-SOA*, es analizar las actuales herramientas, que permitan la integración de este tipo de servicios en el entorno *web*, con las características y funcionalidades aportadas por la semántica.

Problema de investigación.

El desarrollo de servicios *web* para para los procesos de gestación y aprobación de programas de maestrías y especialidades en la actividad de postgrado de la UCLV requiere un alto nivel de integración que no se logra con las herramientas estándares de desarrollo de la *web*.

Objetivo general.

Incorporar semántica para el descubrimiento de servicios *web*, mediante el uso de modelos ontológicos, para los procesos de gestación y aprobación de programas de postgrado en la UCLV.

Objetivos específicos.

1. Seleccionar una estrategia para la definición de servicios *web* mediante el *SOA*.
2. Determinar los servicios *web* referentes a los procesos de gestación y aprobación de programas de postgrado en la UCLV.
3. Determinar el modelo ontológico a utilizar en la incorporación de la semántica.
4. Especificar las herramientas de implementación y ejecución.
5. Incorporar descriptores semánticos a los servicios implementados mediante el modelo seleccionado.

Preguntas de investigación.

1. ¿Qué características se tomarán en cuenta para seleccionar el modelo adecuado para describir la semántica?
2. ¿Cuál soporte tecnológico sería el idóneo para la implementación de servicios *web* semánticos?

Hipótesis.

La extensión con modelos ontológicos de una arquitectura orientada a servicios permitirá la selección, integración, descubrimiento e invocación dinámica de servicios *web* semánticos para los procesos de gestación y aprobación de programa de maestrías y especialidades en la actividad de postgrado de la UCLV.

Capítulo 1: La arquitectura orientada a servicios y los servicios web semánticos.

En el periodo de tiempo transcurrido desde la generalización del uso de la *Web* hasta el momento actual, la riqueza y sofisticación del contenido disponible para los usuarios de esta *Web* se ha incrementado. El salto de complejidad que media desde la concepción de la *Web* como un conjunto de documentos *HTML* enlazados entre sí por hipervínculos, hasta la aparición de las primeras aplicaciones *web* o la construcción de las muchas redes sociales disponibles hoy en día, ha supuesto un desafío constante para las personas involucradas en el desarrollo de las tecnologías *web* que han hecho posible tal evolución.

1.1 Arquitectura orientada a servicios.

La arquitectura orientada a servicios ha sido tratada y definida por varios autores, en su mayoría por empresas y corporaciones internacionales encargadas del desarrollo de la ingeniería de software; cuya primera definición fue dada en 1996, por la empresa consultora y de investigación de las tecnologías de la información *Gartner Group*, con sede en Estados Unidos. Esta la define de la siguiente forma:

"…es una arquitectura de software que comienza con una definición de interface y construye toda la topología de la aplicación como una topología de interfaces, implementaciones y llamados a interfaces. Sería mejor llamada "arquitectura orientada a interfaces". SOA es una relación de servicios y consumidores de servicios, ambos suficientemente amplios para representar una función de negocios completa"(Roche_Benitez, 2015)

SOA establece un marco de diseño para la integración de aplicaciones independientes de manera que desde la red pueda accederse a sus funcionalidades, las cuales se ofrecen como servicios. La forma más habitual de implementarla es mediante servicios *web*, una tecnología basada en estándares e independiente de la plataforma, con la que *SOA* puede descomponer aplicaciones consistentes en un conjunto de servicios e implementar esta funcionalidad en forma modular.(MS, 2006)

Otra de las definiciones principales sobre *SOA* es la dada por el consorcio internacional *World Wide Web* (*W3C*) (W3C, 2004), quien la define como:

"Conjunto de componentes que pueden ser invocados, cuyas descripciones de interfaces se pueden publicar y descubrir"

En lo referente a la aceptación de la anterior definición, el Instituto Federal Estadounidense de Investigación y Desarrollo, encargado de desarrollar modelos de evaluación y mejora en el desarrollo de software, denominado *Software Engineering Institute* (*SEI*); ha rechazado su supuesto presentando como argumento que ésta sobreestima el papel de los componentes de *SOA*, no considerando el rol de la práctica y construcción de la arquitectura como partes importantes de su proceso de conformación.

Otra definición cardinal referida a la Arquitectura Orientada a Servicios es la expuesta por la corporación multinacional de computadoras, tecnología y consultoría de *IT*, *IBM* (*International Business Machines*), quien la define como: *"Una arquitectura de aplicación en la cual todas las funciones se definen como servicios independientes con interfaces invocables bien definidas, que pueden ser llamadas en secuencias definidas para formar procesos de negocios"* (Roche_Benitez, 2015)

Se hace notar en esta conceptuación el hecho de que *SOA* no tiene una única finalidad, sino que ésta proporciona un marco de reutilización de sus componentes, con lo cual se infiere que ésta sirve como base de ejecución para disimiles aplicaciones.

Una arquitectura técnica basada en *SOA* consiste, según Erl (2005), de tres capas, tal como se muestra en la Figura 1.1:

- Una capa de gestión de procesos de negocio, que coordina la ejecución de servicios de negocio;
- Una capa de servicios de negocio, que entrega los servicios de información a los procesos de negocio; y
- Una capa de servicios de aplicación de negocio, que ejecuta la lógica de la aplicación y se encarga del almacenamiento de los datos.

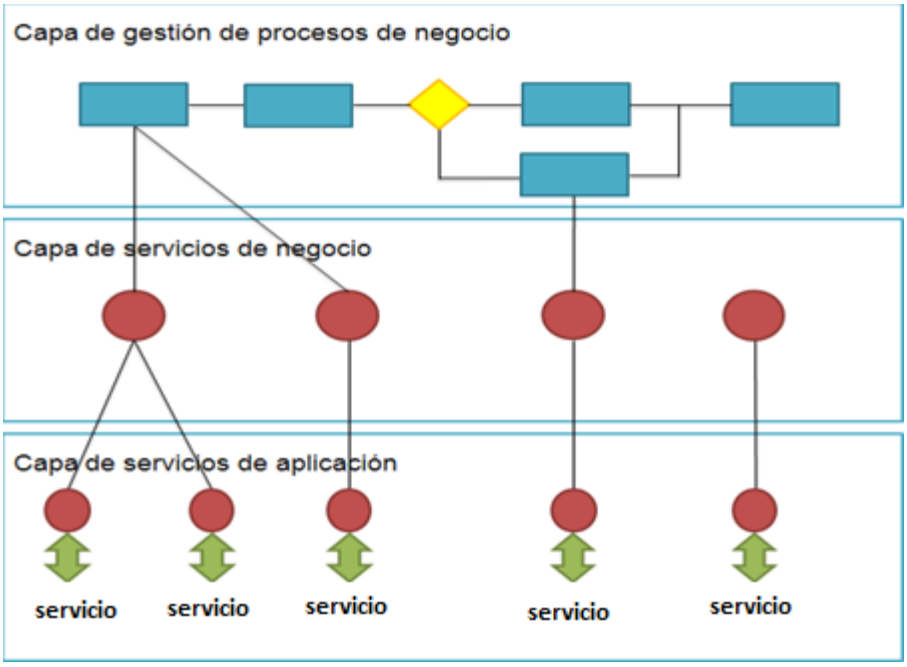


Figura 1.1: Capas de servicio de una arquitectura orientada a servicio.

La modularidad de los componentes de software y la interoperabilidad abierta que ofrece *SOA*, ha permitido crear diversas soluciones altamente escalables.

1.1.1 Beneficios de una arquitectura orientada a servicio.

Los beneficios de *SOA* para una organización se plasman a dos niveles distintos, al del usuario corporativo y a nivel de la organización de IT.

Desde el punto de vista de la empresa, *SOA* permite el desarrollo de una nueva generación de aplicaciones dinámicas que resuelven una gran cantidad de problemas de alto nivel, fundamentales para el crecimiento y la competitividad. Las soluciones *SOA* permiten entre otras cosas:

- Mejorar la toma de decisiones
- Mejorar la productividad de los empleados
- Potenciar las relaciones con clientes y proveedores
- Aplicaciones más productivas y flexibles
- Desarrollo de aplicaciones más rápido y económico
- Aplicaciones más seguras y manejables

SOA contribuye también a documentar el modelo de negocio de la empresa y a utilizar el modelo de negocio documentado para integrar en él y dar respuesta a las dinámicas de cambio que se produzca y optimizarlo de acuerdo con ellas.(MS, 2006)

1.1.2 Estrategias de desarrollo de una arquitectura orientada a servicios.

La estrategia debe basarse en las prioridades de una organización para establecer el balance correcto entre la entrega de metas de migración de largo plazo con el cumplimiento de requisitos de corto plazo. Tres estrategias comunes han emergido, cada una ocupándose de este problema de una manera diferente (Ramollari *et al.*, 2007).

De arriba hacia abajo.

La estrategia de arriba hacia abajo soporta la creación de las tres capas de servicios. Es común que este acercamiento resulte en la creación de numerosos negocios reutilizables y aplicaciones de servicios. La estrategia de arriba hacia abajo típicamente contendrá una cierta cantidad o todos los pasos ilustrados y descritos en la figura 1.2. Este proceso da por supuesto que los requisitos del negocio ya han sido coleccionados y definidos.

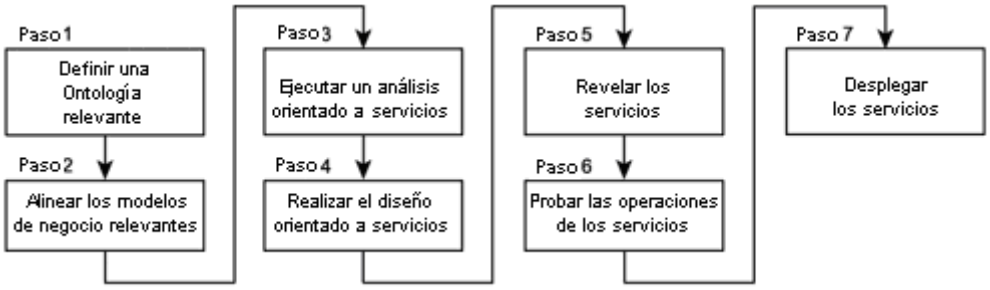


Figura 1.2: Estrategia de desarrollo de arriba hacia abajo.

El enfoque de arriba hacia abajo en la construcción de *SOA* generalmente resulta en una arquitectura de servicios de alta calidad. El diseño y los parámetros en torno a cada uno de los servicios se analizan a fondo, la maximización de la reutilización potencial y las oportunidades para optimizar las composiciones. Los obstáculos a raíz de un enfoque de arriba hacia abajo por lo general están asociados con el tiempo y dinero. Las organizaciones están obligadas a invertir significativamente en el análisis inicial de los proyectos que pueden tener una gran cantidad de tiempo sin resultados inmediatos.

De abajo hacia arriba

Este acercamiento esencialmente promueve la creación de servicios como una manera de satisfacer los requisitos de la aplicación. Los servicios *web* son implementados "como sean necesarios" y modelados para encapsular la lógica de la aplicación sirviendo mejor a los requisitos inmediatos de la solución. La integración es la motivadora primaria para diseños de abajo hacia arriba, dónde la necesidad de tomar ventaja del marco de trabajo abierto de comunicaciones *SOAP* puede ser encontrada en el simple hecho de anexar servicios como envolturas para sistemas legados. Un típico enfoque de abajo hacia arriba sigue un proceso parecido a lo uno explicado en la Figura 1.3. Este proceso da por supuesto que los requisitos del negocio ya han sido coleccionados y definidos.

La mayoría de las organizaciones que actualmente están construyendo los servicios *web* aplican el enfoque de abajo hacia arriba. La razón principal detrás de esto es que las organizaciones simplemente pueden añadir servicios *web* a sus entornos de aplicaciones existentes para aprovechar esta tecnología. La arquitectura de servicios *web* y los servicios de orientación en el que se añaden se mantiene sin cambios, por lo tanto, los principios son raramente tenidos en cuenta.

Aunque el diseño de abajo hacia arriba permite la creación eficiente de los servicios *web* requerido por las aplicaciones, la implantación de una arquitectura *SOA* en un momento dado puede necesitar una gran cantidad de reajustes, o incluso la introducción de nuevas capas de servicios.

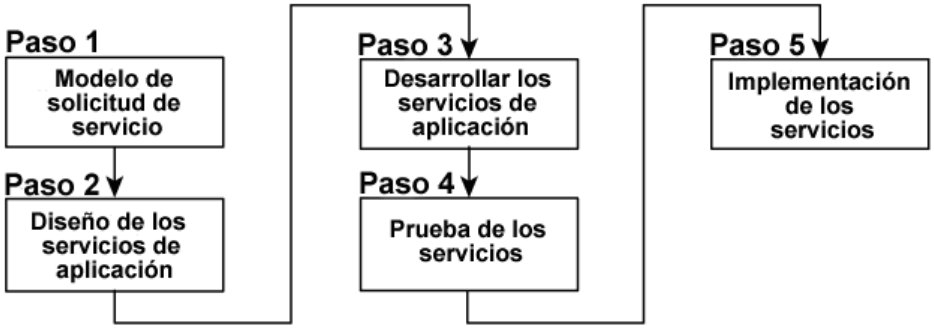


Figura 1.3: Estrategia de desarrollo de abajo hacia arriba

Estrategia Ágil

Para muchas organizaciones es útil mirar las dos estrategias anteriores como extremas y encontrar un adecuado intermedio. Esto es posible definiendo un nuevo proceso que tiene en cuenta el análisis de los niveles de negocio ocurriendo concurrentemente con diseño de servicio y desarrollo. También conocido como satisfacer-en-el-medio, la estrategia ágil es más complicada que las dos

previas, simplemente porque necesita satisfacer dos conjuntos opuestos de requisitos. Los pasos del proceso se muestran en la Figura, demuestran un ejemplo de cómo una estrategia ágil puede usarse para alcanzar las metas de los enfoques de arriba hacia abajo y de abajo hacia arriba respectivamente.

Esta estrategia toma lo mejor de las dos anteriores, sin poner en peligro la integridad del modelo de negocio de una organización y los servicios orientados a las características de la arquitectura.

A pesar de que cumple tanto a corto como a largo plazo, el resultado neto del empleo de esta estrategia es aumentar los esfuerzos asociados a la prestación de cada servicio.

Este enfoque impone tareas de mantenimiento que se requieran para garantizar que los servicios existentes se mantengan en consonancia con los modelos de negocio revisados. Incluso con un proceso de mantenimiento en su lugar, es difícil lograr la alineación con un modelo de negocio que se encuentra en constante cambio (Figura 1.4).

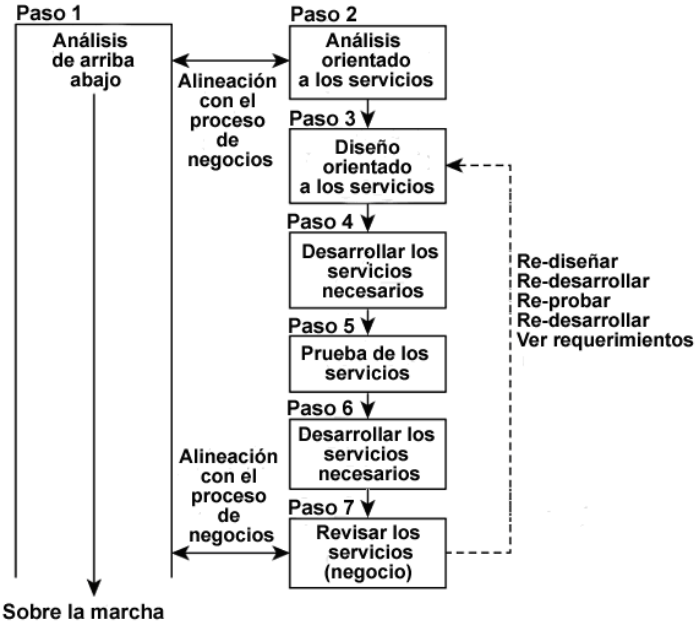


Figura 1.4: Estrategia de desarrollo ágil

1.2 Servicios *Web*. Facilitador tecnológico de la arquitectura orientada a servicios.

Para proporcionar un sistema de alta disponibilidad que contribuya a una mayor integración y comunicación entre el software y su futura aplicación, se hace necesaria la utilización de herramientas que garanticen la interacción entre los sistemas y sus aplicaciones, de modo que el diseño de su propia estructura contemple la posibilidad de facilitar cambios a posteriori.

La adopción de una solución de diseño basada en *SOA* no exige necesariamente implantar servicios *web*; no obstante, los servicios *web* son la forma más habitual de implementar *SOA*.

Los servicios *web* son aplicaciones que utilizan estándares para el transporte, codificación y protocolo de intercambio de información. Permiten la intercomunicación entre sistemas de cualquier plataforma y se utilizan en una gran variedad de escenarios de integración, tanto dentro de las organizaciones como con *partners* de negocios (Valledor_Pellicer, 2006).

Entre las características de los servicios *web*, se pueden destacar las siguientes (Krishnamurthi *et al.*, 2005):

- Proporcionan interoperabilidad, es decir, varias aplicaciones con propiedades completamente diferentes, desarrolladas mediante lenguajes de programación distintos e independientemente del fabricante y de la plataforma sobre la cual se hayan instalado, pueden comunicarse en una red de ordenadores mediante los servicios *web*.
- Además superan las barreras geográficas, ofreciendo la posibilidad de integrar servicios de compañías diferentes y distantes geográficamente.
- Proporcionan una gran flexibilidad de modo que no supone un problema grande que se realicen cambios en las aplicaciones que usan los servicios.
- Se basan en *HTTP* sobre *TCP* en el puerto 80. Se permite también que la comunicación sea segura vía *SSL* sobre *HTTPS*.
- Permiten a los programadores abstraerse de los procesos de negocio y centrarse en el entorno de programación de los servicios *web*.

1.2.1 Tecnologías y estándares de servicios *web*.

El servicio *web* realiza sus disímiles funcionalidades a través de cuatro elementos fundamentales, mediante los cuales se hacen posibles las intercomunicaciones e intercambios de datos, que éste ofrece como facilidades de uso. Estos elementos son los siguientes: el *eXtensible Markup Language (XML)*, el *Web Services Description Language (WSDL)*, el *Simple Object Access Protocol (SOAP)* y el *Universal Discovery Description and Integration (UDDI)* (Roche_Benitez, 2015).

- *eXtensible Markup Language* (*XML*): El *XML* fue diseñado fundamentalmente para ejecutar la descripción de datos en los documentos *Web*. La principal ventaja que éste ofrece es que

permite a los diseñadores crear sus propios *tags*, facilitando con ello una más óptima modificación, definición, transmisión, validación e interpretación de datos entre aplicaciones. De modo que se presenta como un lenguaje con altos grados de flexibilidad, que presta más atención al contenido de la información.

- *Web Services Description Language* (*WSDL*): El *WSDL* se conforma en un protocolo, basado fundamentalmente en *XML*, mediante el cual se presenta los accesos al *Web Services*. Generalmente puede ser comprendido como un manual de operaciones que indica qué interfaces provee el *Web Services*, así como cuáles son los tipos de datos necesarios para la utilización del mismo.

- *Simple Object Access Protocol* (*SOAP*): *SOAP* es un protocolo basado en *XML*, y es principalmente el responsable del sistema de comunicación de base existente entre los *Web Services*. Tiene como característica propia que es independiente de plataforma y de lenguaje, por lo que ofrece un amplio espectro de posibilidades de adaptación.

- *Universal Discovery Description and Integration* (*UDDI*): El *UDDI* conforma un modelo de directorios para los Servicios *web*, cuya función es mantener estandarizada en un formato universal la información que se maneja dentro del mismo, así como sus capacidades, requerimientos y ubicación. De modo general, el *UDDI* (Figura 1.5) permite conocer cuáles son los servicios que ofrece el servicio *web* en cuestión.

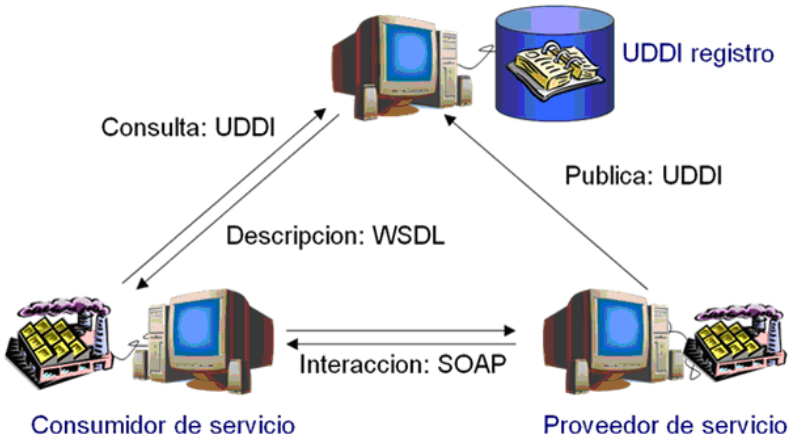


Figura 1.5. Descubrimiento e invocación de servicios *web*.

1.2.2 Herramientas.

Altova: Es un editor de *XML* y entorno de desarrollo integrado o *IDE* (siglas en inglés para *Integrated Development Enviroment*) desarrollado por *AltovaXML*Spy. Es el editor *XML* más vendido gracias a sus avanzadas funciones de modelado, edición, transformación y depuración de tecnologías *XML*. Ofrece el diseñador gráfico de esquemas líder de la industria, generador de código, conversores de archivos, generadores de perfiles, compatibilidad con BD, *XSLT*, *XPath*, *XQuery*, *WSDL*, *SOAP*, *XBRL*, JSON y *OOXML*, además de una perfecta integración con Visual Studio y Eclipse (*AltovaXML*, 2016).

NetBeans: Entorno de desarrollo integrado o *IDE*, libre, de código abierto, multiplataforma con soporte integrado para el lenguaje de programación Java. La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos.

Soporta el desarrollo de todos los tipos de aplicación Java (J2SE, *web*, *EJB* y aplicaciones móviles). Permite crear aplicaciones *web* con *PHP* 5, un potente *debugger* integrado y además viene con soporte para *Symfony* un gran *framework* MVC escrito en *php*.

Provee soporte para la creación de aplicaciones orientadas a servicios (*SOA*), incluyendo herramientas de esquemas *XML*, un editor *WSDL*, y un editor *BPEL* para *web services* (ORACLE, 2016).

Eclipse: Esta plataforma es usada para desarrollar entornos de desarrollo integrados (del inglés *IDE*) que se pueden utilizar para crear aplicaciones tan diversas como sitios *web*, programas *Java* embebidos, programas de C ++, y *JavaBeans Enterprise* (Eclipse, 2003).

Apache Axis: Es un motor para servicios *web*. En agosto del 2004 fue lanzado con un rediseño total y una reimplementación completa de la ampliamente difundida pila *SOAP* "Apache Axis", no solo provee la capacidad de agregar servicios *web* a las aplicaciones *web*, sino que además puede funcionar como servidor autónomo. Se desarrolla bajo los auspicios de la *Apache Software Foundation. Apache Axis2* es más eficiente, modular y más orientado a *XML* que la versión anterior y extiende su funcionalidad a través de la fácil adición de módulos *plugins* (ASF, 2016).

Protégé: Es un editor libre de código abierto y un sistema de adquisición de conocimiento. Al igual que *Eclipse*, *Protégé* es un *framework* para el cual otros proyectos sugieren *plugins*, como es el

plugin OWL-S (*Semantic for Ontological Web Language*), el cual se basa en el modelo ontológico del mismo nombre para la edición de las instancias de las clases y parámetros necesarios en la incorporación de la semántica (BMIR, 2016).

1.3 Servicios web semánticos.

Mediante la especificación sintáctica de los servicios *web*, es imposible automatizar procesos tales como descubrimiento, ejecución y composición de servicios. Para solucionar este problema, se desarrollaron los *SWS*, que consisten en la fusión de los servicios *web* tradicionales y de las tecnologías empleadas en la *web* semántica, que permiten a las máquinas interpretar la información que almacenan, empleando las ontologías como modelo de datos, como se aprecia en la figura 1.6 (Valledor_Pellicer, 2006).

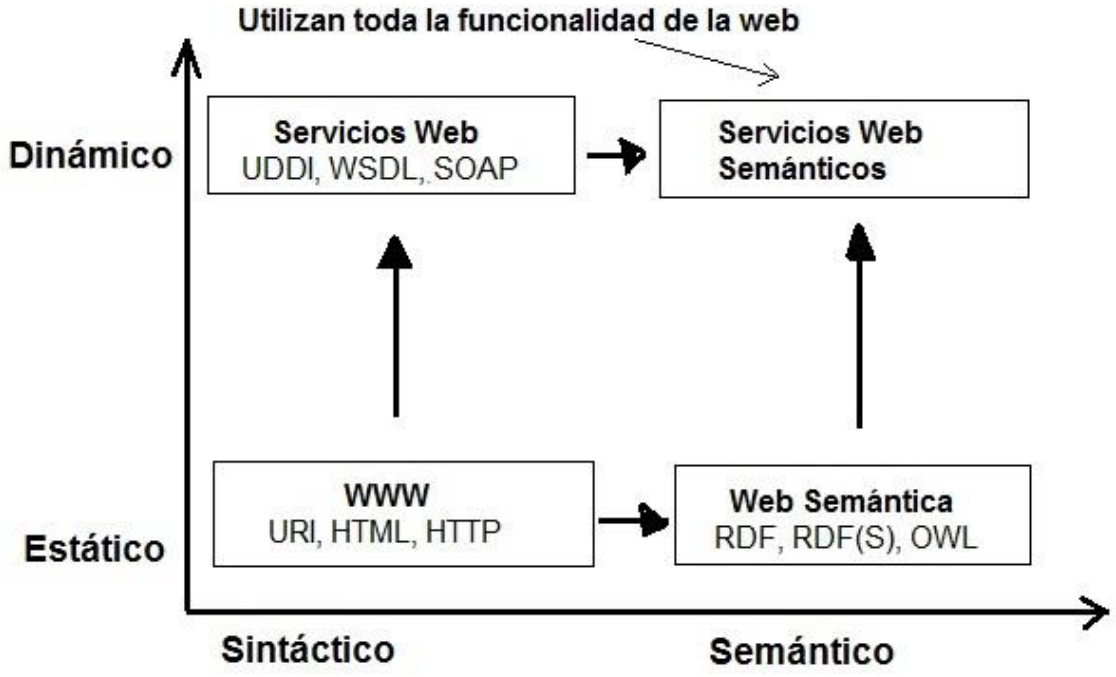


Figura 1.6: Esquema de las tecnologías participantes en los *SWS*.

1.3.1 La *Web* Semántica o *Web* 3.0.

El actual entorno digital es más participativo, extenso e interconectado y lleva a una nueva organización de los conocimientos más colaborativa (folksonomías), más interconectada (ontologías), más interactiva (interfaces persona-ordenador), más realidad virtual, aumentada,

inmersiva e interior (herramientas tecnológicas), más participativa (contenidos en blogs, wikis, redes sociales). Ya no son válidas las tradicionales descripciones para organizar la *web*, sino cómo operan los que acceden a la misma (San_Segundo *et al.*, 2012).

La *Web* 3.0 es uno de los temas de máxima actualidad. (Berners-Lee's, 2004) intentó al inicio de la *web* incluir informaciones semánticas en su creación, por diferentes motivos no ha sido posible e introdujo el concepto de semántica con la intención de recuperar dicha omisión. El futuro de la *web* depende de cómo se organice la *web* semántica, esta ha de ser interoperable sin estructuras previas de jerarquización y exclusión, una *web* asociativa y colaborativa, para relacionar cualquier tipo de información, de formato y tipo que fuere integrado la ya denominada inteligencia digital colectiva.

1.3.2 Evolución tecnológica hacia servicios *web* semánticos.

A partir de los esfuerzos de estandarización del *W3C* y a través de su grupo de interés *Semantic Web Services Interest Group* se está fomentando la integración de la tecnología de la *Web* Semántica y el trabajo de otros grupos sobre servicios *web* desarrollado en *W3C*. Una de las iniciativas más importantes surgidas a partir de este trabajo ha tomado como base el trabajo desarrollado por la coalición *DAML-S* (*DAML for Services*), surgida en el año 2001, la cual propuso una ontología para la descripción semántica de servicios basada en *DAML+OIL* (*DAML-S*), cuya última versión (0.9) fue lanzada en mayo de 2003. A partir de noviembre de 2003 se desarrolló *OWL-S*, basado en el lenguaje de marcado ontológico *OWL*. Actualmente esta iniciativa es el resultado de la unión de otras dos: *Semantic Web Services Initiative* (*SWSI*) y *Semantic Web Services Language (SWSL) Committee*. *OWL-S* fue enviado a *W3C* para su reconocimiento en julio de 2004 y en el momento de escribir estas líneas, todavía no supone una recomendación (Cabral *et al.*, 2004) .

Existen diversas tecnologías involucradas en la evolución de los servicios a servicios *web* semánticos. En la figura 1.7 se puede ver esta evolución, así como las principales fechas que recogen algún momento importante, como el lanzamiento de una determinada tecnología, el envío a *W3C* o finalmente la recomendación o estandarización por parte de este consorcio. Sin embargo, hay que aclarar que las fechas que aparecen en la figura no siguen un orden cronológico en esta evolución, ya que a veces, las recomendaciones son posteriores al uso de estas tecnologías o simplemente las fechas reflejan el lanzamiento de versiones posteriores. Es importante, destacar el

continuo y dinámico trabajo llevado a cabo en cada uno de las áreas, lo que demuestra el gran esfuerzo que se lleva realizando así como el gran interés suscitado (Samper_ Zapater, 2005).

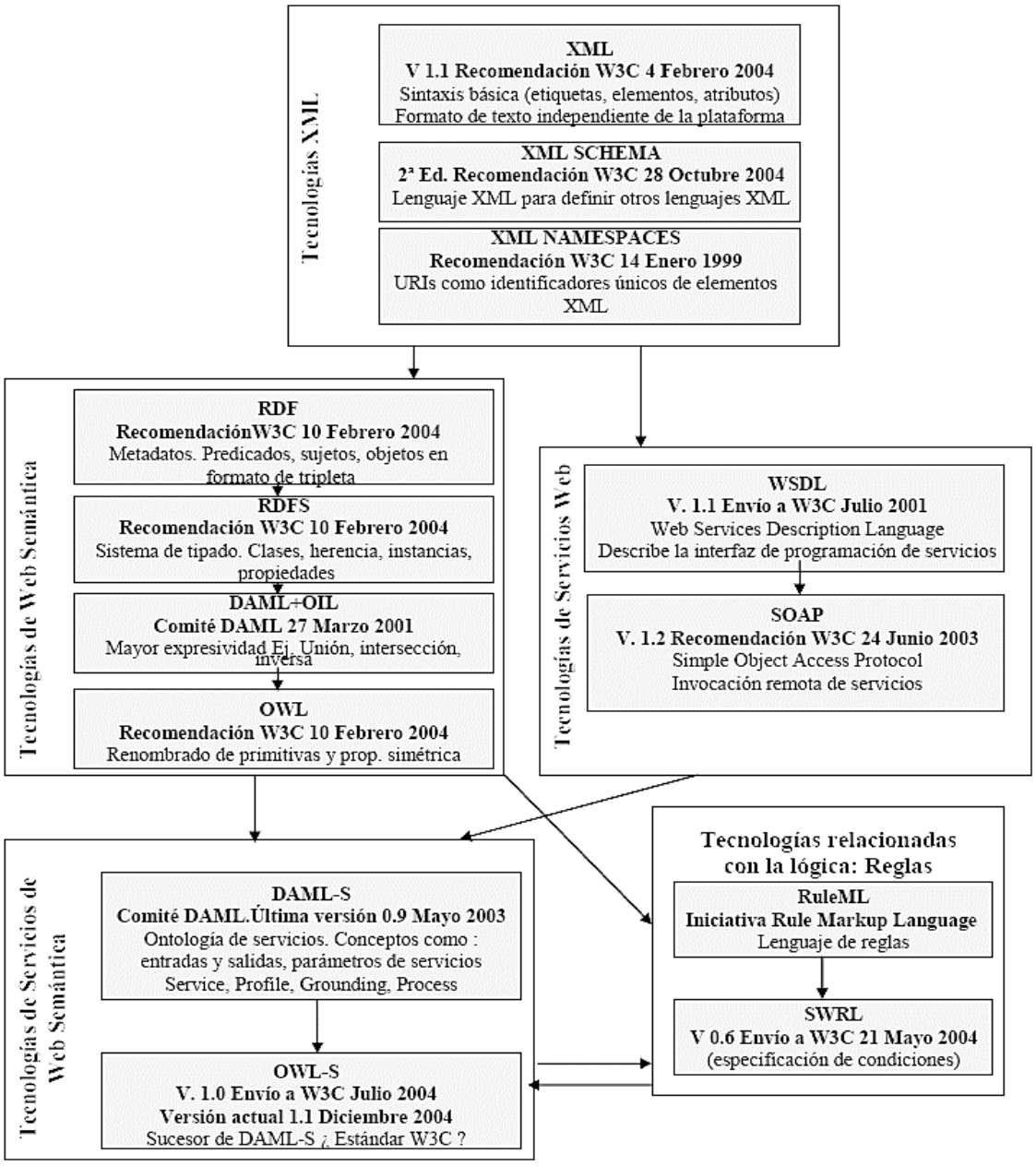


Figura 1.7: Tecnologías involucradas en servicios *web* semánticos

Mientras que lenguajes como *WSDL* son los encargados de aspectos de comunicación de bajo nivel para los servicios de la *Web*, *DAML-S* fue desarrollado para proporcionar una capa de aplicación de alto nivel por encima de éstos. Como estos lenguajes de comunicación de bajo nivel y protocolos definen "cómo" tener acceso a un servicio de *web*, tanto *DAML-S* como su predecesor *OWL-S*

tienen como objetivo el proporcionar una respuesta en cuanto a "por qué" se utiliza un cierto servicio, y lo que este servicio hace realmente (Sollazzo *et al.*, 2002).

El requerimiento de expresividad semántica, añaden también el hecho de que las descripciones de servicios permitan realizar comparaciones de manera automática, y por tanto que los servicios tengan una cierta estructura para lograr este objetivo, además de los requerimientos de flexibilidad y edición. La búsqueda de servicios mediante descripciones dadas por estos lenguajes, a veces no se puede llevar a cabo mediante plantillas fijas o comunes y por tanto el lenguaje debe ser lo suficientemente flexible como para permitir tal expresividad (Varga *et al.*, 2005).

La conveniencia de que las descripciones de servicio deban ser creadas, leídas y editadas por humanos establece el último de los requerimientos. Se establecen los requerimientos para el descubrimiento de un servicio *web*, para cada una de las partes que forman parte del proceso: descripción del servicio, publicación, descripción de la petición, y por último el servicio de emparejamiento.

Otro aspecto importante argumentado anteriormente, al describir diferentes casos de uso, es la distinción entre peticiones de servicios volátiles y persistentes. Las primeras serían aquéllas en las que el *matchmaker* devuelve en el instante del requerimiento aquellos anuncios compatibles con la petición, mientras que las segundas se corresponden con peticiones que pueden ser respondidas en un tiempo posterior y dentro de un periodo de validez definido previamente, tras la incorporación de nuevos anuncios o modificación de los ya existentes. Por último afirman la necesidad de dar soporte a la composición de servicios, debido a que posiblemente determinados servicios no puedan satisfacer una determinada petición por si solos, pero sin embargo una combinación de éstos sí (Samper_ Zapater, 2005).

1.3.3 Infraestructura de los servicios web semánticos.

La infraestructura de los *SWS*, se puede caracterizar en tres dimensiones: actividades de uso, arquitectura y ontología del servicio (Valledor_Pellicer, 2006).

- **Las actividades de uso del *SWS*** identifican los requisitos funcionales que un framework de servicios *web* semánticos debe soportar.

Las principales actividades son:

- Publicación de servicios en un registro semántico.

- Descubrimiento automático en función de la petición de servicio y de la descripción semántica publicada del servicio.
- Selección de servicios cuando se pueda producir un conflicto entre varios servicios que satisfagan una petición.
- Composición: Un *framework* de *SWS* debe permitir la coreografía de servicios, de forma que se puedan componer los resultados de varios servicios para obtener una funcionalidad de un nivel más alto. Un lenguaje de descripción de procesos, empleado para la composición de servicios, podría ser el *BEPL4WS*.
- Invocación del *SWS*: validación de los parámetros de entrada contra la ontología del servicio y ejecución del servicio.
- Despliegue
- Gestión de las ontologías

❖ **La arquitectura de un framework de *SWS*** define los componentes necesarios para cumplir las actividades de uso especificadas anteriormente.

Estos componentes son:

- Un registro que ofrezca los mecanismos necesarios para publicar y localizar los *SWS*.
- Un razonador para las consultas e interpretación de la descripción semántica de los servicios.
- *Matchmaker*: Mediador entre el registro y el cliente del servicio en los procesos de descubrimiento y selección de servicios.
- *Decomposer*: Componente encargado de ejecutar la composición de servicios. Parte de la semántica de la petición realizada, tratando de descomponerla, de forma que se busca la colaboración de las funcionalidades ofrecidas por varios servicios para obtener el resultado deseado.
- Invocador: Mediador entre el cliente del servicio y el proveedor o entre el decompositor y el proveedor cuando se invoca un servicio.

❖ **La ontología del servicio** representa las capacidades del servicio y sus restricciones de uso. Integra la semántica del servicio con su descripción.

Consta de los siguientes elementos:

- Información funcional del servicio
  - Entradas
  - Salidas
  - Precondiciones
  - Postcondiciones
- Información no funcional
  - Categoría
  - Coste
  - Calidad de servicio

En general, los *SWS* son una línea importante de la *web* semántica, que propone describir no sólo la información de acceso a un servicio, sino definir vocabularios de funcionalidad y procedimientos para describir servicios *web*. Tal descripción abarca aspectos como entradas, salidas, procesos, condiciones necesarias para que se puedan ejecutar, los efectos que producen y la información para localizarlos. En la Figura 1.8 podemos apreciar la arquitectura de la *web* semántica y el papel que desempeñan los *SWS* dentro de ella.

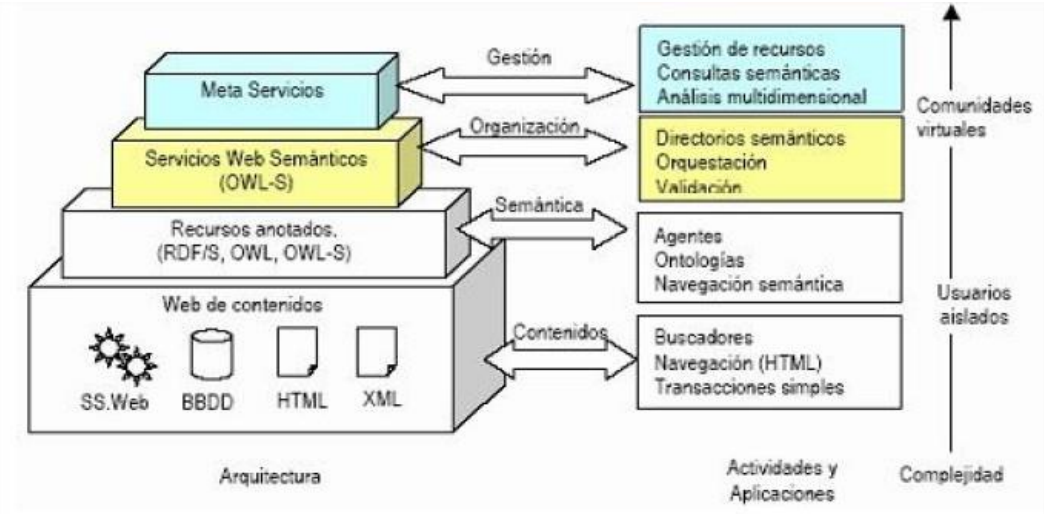


Figura 1.8: Arquitectura de la *web* semántica.

1.4 Ontologías.

Las ontologías son el concepto núcleo de la *web* semántica, que permite representar la información de manera procesable por un computador. En el ámbito de las tecnologías de la

información y la comunicación, estas han suscitado un gran interés en los últimos años, sobre todo a raíz de la consideración por parte del grupo *W3C* como la tecnología que está llamada a proporcionar la infraestructura de conocimiento a la *web* semántica.

Existen variadas definiciones tecnológicas para el concepto de ontología como son las citadas por Fernández_Brise (2008), Falcón_Espinosa (2009), Michán (2011), Caliusco (2011), quienes coinciden, como otros tantos autores, en mencionar la concisa pero efectiva definición de Gruber (1993)*: "Una ontología es una especificación explícita y formal de una conceptualización […], lo que existe es lo que puede ser representado. Cuando el conocimiento de un dominio es representado en un formalismo declarativo, el conjunto de objetos que pueden ser representados es llamado el universo del discurso […], las definiciones asocian nombres de entidades del universo del discurso con textos comprensibles por los humanos que describen el significado de los nombres, y axiomas formales que limitan la interpretación y buen uso de dichos términos. Formalmente, una ontología es una teoría lógica".*

La razón por la cual las ontologías se han convertido en tan populares es, en gran medida, debido a lo que prometen: una comprensión compartida y común de algún dominio que puede ser comunicado entre individuos y aplicaciones. Aunque su aplicación empezó en el área de la inteligencia artificial, las ontologías se han convertido en un tema de investigación importante en diferentes áreas como la ingeniería y la representación del conocimiento, el procesamiento del lenguaje natural, la integración inteligente de información, los sistemas cooperativos de información, la recuperación de información, el comercio electrónico, la gestión de conocimiento, etc. Debido a que las ontologías pretender ser conocimiento consensuado del dominio, su desarrollo es frecuentemente un proceso cooperativo que implica a diferentes individuos, posiblemente en diferentes situaciones geográficas.

1.4.1 Principales características de las ontologías.

Las definiciones de una ontología deben ser objetivas, es decir, independientes del contexto que motivaron su definición. Todas las definiciones deben documentarse con lenguaje natural (Caliusco, 2011).

Una ontología facilita la explicitación específica de los conceptos, sus propiedades y sus restricciones, así como la organización parcial de estos conceptos a partir de sus relaciones. Por ello, no puede interpretarse que los conceptos se organizan, exclusivamente, de forma jerárquica

como en una taxonomía (por medio de las relaciones taxonómicas "*es-un"* y "*parte-de*"). En función del dominio pueden existir otros tipos de relaciones semánticas que pueden determinar organizaciones de tipo grafo o redes, por ejemplo, en el área de la salud, relaciones del tipo "afecta-a", "trata-a", "ingrediente-de", "reacciona-con" (Romá_Ferry, 2009).

La ontología, necesariamente, incluirá un vocabulario de términos y una especificación de sus significados para restringir las posibles interpretaciones. Según Gruber (1993), el conocimiento en ontologías se formaliza principalmente mediante el uso de cinco tipos de componentes: conceptos o clases, relaciones, funciones, axiomas e instancias.

Conceptos: Son las ideas básicas que se intentan formalizar. Pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento, etc. El conjunto de conceptos identificados es denominado universo del discurso y representan el conocimiento de un dominio a través de un formalismo declarativo.

Relaciones: Establecen el tipo de interacción semántica entre los conceptos, de manera que no solo se hace explícito lo que denotan, sino también lo que connotan.

Funciones: Constituyen un tipo especial de relación, a través de las cuales se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como categorizar-clase, asignar-fecha, etc. Es imprescindible en ontologías para modelar sistemas y procesos.

Axiomas: Son teoremas declarados sobre relaciones que deben cumplir los elementos de la ontología. Permiten hacer inferencias que no están explícitas en los conceptos.

Instancias: Contienen elementos o datos que describen o ejemplifican un concepto. Por lo general son propiedades que no se recogen en el concepto o nociones que sirven para aclarar la connotación de este en un contexto determinado.

Existen autores que atribuyen otras cantidades de componentes según su aplicación o tipología.

En la literatura sobre aplicaciones tecnológicas de ontologías pueden encontrarse diferentes tipos de estas últimas. Se usan dos criterios principalmente para tales clasificaciones: (a) el tipo de conocimiento contenido; y (b) la motivación de la ontología.

1.4.2 Construcción y aplicación de ontologías como productos intermedios.

Crear una ontología conlleva realizar elecciones y seleccionar criterios y, puesto que el objetivo principal es de valer como referencia a personas y aplicaciones, es necesario que estas elecciones y categorizaciones estén consensuadas y aceptadas entre los individuos. De este modo, las ontologías son descripciones conceptuales y terminológicas de un entendimiento compartido sobre un dominio específico (Moreno_Ortiz, 2005).

La aplicación de una ontología determina en muchos casos la categorización que se hace. Una clasificación o taxonomía se hace siempre desde dominio específico y con un punto de vista determinado, es decir, teniendo un criterio que determina la jerarquía y las relaciones que se definen en el dominio.

Ser una fuente de conocimiento es ya, en sí misma, una aplicación de una ontología, pues se pueden extraer los conceptos, definiciones, clasificaciones, restricciones etc., consensuados en su construcción. Como artefacto o requerimiento de software, una ontología siempre será muy útil a la hora explotar ese conocimiento, ya que sirven para traducir los términos usados por una aplicación a otra, estas suelen estar escritas en lenguajes de programación diferentes. La posibilidad de su reutilización la hace merecedora de tantas funciones como productos intermedios en la ingeniería y representación del conocimiento, en la creación de sistemas de información cooperativos, gestión del conocimiento y razonamiento automático, etc.

1.5 Modelos ontológicos para servicios web semánticos.

Existen en la actualidad tres *frameworks* de *SWS*: *OWL-S, WSMO y SAWSDL* (Valledor_Pellicer, 2006, Varga *et al.*, 2005, Sycara *et al.*, 2006, Stollberg *et al.*, 2006).

- **Modelo *OWL-S*.**

Es una ontología de servicios *web* basada en *OWL*. Añade un marcado de metadatos en los proveedores de los servicios *web*, permitiendo la descripción de las propiedades y funcionalidades de cualquier servicio, de forma no ambigua, bien definida y computable.

Utiliza como metalenguaje, para especificar modelos de conceptos, *DAML+OIL*, lenguaje de marcado a partir del cual ha surgido el *OWL*, que es empleado como base para el desarrollo de las ontologías de *OWL-S* (Figura 1.9).

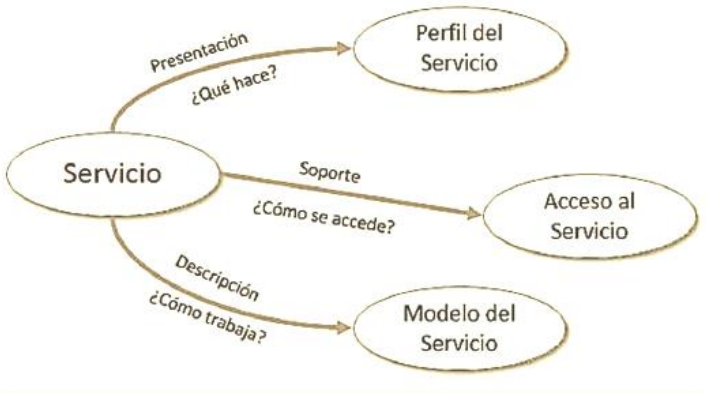


Figura 1.9: Modelo ontológico *OWL-S* para la descripción de servicios web.

La ontología propuesta por *OWL-S* está formada por tres partes o clases fundamentales:

- **Perfil del servicio**, empleado para el descubrimiento y selección de servicios.

  El perfil de cada servicio consta de los siguientes elementos:

  - Información sobre el proveedor del servicio: nombre del servicio, descripción textual breve, información de contacto.
  - Características del servicio (propiedades): categoría del servicio y parámetros de libre disposición.
  - Funcionalidad del servicio: Transformación realizada entre los datos de entrada y de salida, precondiciones y efectos del servicio. Es decir, se describe el *IOPE* (*Inputs-Outputs-Preconditions-Efects*) del servicio.

- **Modelo del servicio**, ontología encargada de describir con detalle las operaciones procesadas por un servicio. El servicio se modela como un proceso de interacción entre el cliente y el servicio. Existen tres tipos de proceso:

  - Atómico: Formados por *IOPE* (Entradas, salidas, precondiciones y efectos). Pueden ser ejecutados directamente.
  - Compuesto: Tienen asociados una estructura de control (Secuencia, Condición, Iteración, Repetir-Hasta…)
  - Simple: Abstracción de procesos atómicos y compuestos.

- **Acceso al servicio**, conocimiento base, que describe como interoperar con el servicio, a través de mensajes. El conocimiento base especifica cómo acceder e invocar al servicio. Realiza un descripción de las relaciones existentes entre la descripción del proceso realizado por el servicio y la descripción real de los elementos necesarios para interaccionar con el servicio (como los protocolos de comunicaciones).

En resumen, cada servicio posee un conjunto de perfil del servicio, de forma que pueda proporcionar diferentes funcionalidades a los clientes. El descubrimiento automático de servicios se hace mediante el *matching* de los perfiles definidos en el sistema. Cada servicio requiere de un único modelo del servicio, que indica su funcionamiento, y se comunica con el cliente mediante un único acceso al servicio.

El estado de desarrollo de entornos basados en *OWL-S* es escaso. Algunas herramientas son:

- *OWL-S* Editor, desarrollado por la Universidad de Malta.
- *OWL-S* Java *API*, desarrollada por *Mindswap*
- *OWL-S* Editor basado en *Protegé*.

❖ **Modelo *WSMO*.**

Es una ontología y un modelo conceptual para describir servicios *web* semánticos.

Está basada en el *WSMF* (*Web Service Modeling Framework*). La idea principal de *WSMO* es la de desarrollar un *framework* de *SWS* que permita resolver todos los problemas posibles en la integración de los *SWS* y de generar una implementación del mismo, capaz de servir como referencia. Todos estos aspectos se han de basar en dos principios:

- Desacomplamiento: Las aplicaciones han de ser lo más independientes posibles.
- Comunicaciones escalables de mediación, permitiendo que todo el mundo pueda intercomunicarse con todo el mundo.

El proyecto *WSMO* se divide en tres partes: *WSMO, WSML* y *WSMX*.(Figura 1.10)

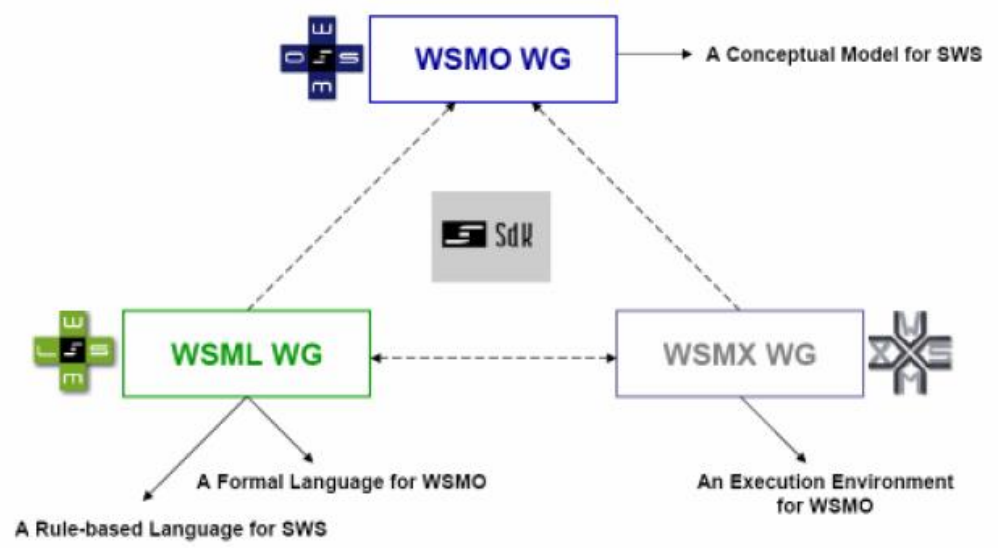


Figura 1.10. Partes que conforman el proyecto WSMO

La ontología desarrollada como *WSMO*, expresada en *WSML* y publicada en el *W3C* como *Member Submission*, se basa en la utilización de cuatro componentes:

- Las ontologías, proporcionan la especificación formal de la información usada por todos los componentes.
- Los objetivos deseados por los clientes cuando invocan un servicio *web*, describen cual es el problema a resolver.
- La semántica de los servicios *web*, describen su interfaz de uso (utilizada para coreografía y orquestación de servicios) y sus funcionalidades (definidas mediante *IOPE*). También se especifican las propiedades no funcionales de los servicios, por ejemplo utilizado *Dublin Core*.
- Por último, los mediadores, son conectores entre componentes, permitiendo fácilmente la integración entre información heterogénea. Resuelven problemas de interoperabilidad y pueden ser de cuatro tipos:
  - Mediadores entre ontologías (*OOMediators*): Resuelven incompatibilidades.
  - Mediadores entre objetivos (*GGMediators*): Buscan el refinamiento y la compatibilidad entre objetivos.
  - Mediadores entre un servicio *web* y un objetivo (*WGMediator*): Busca la satisfacción de las necesidades.
  - Mediadores entre servicios *web* (*WWMediator*): Utilizado en composición de servicios.

WSML

El *Web Service Modeling Language*, es un nuevo lenguaje de modelado de ontologías desarrollado para modelar WSMO. Una de sus características principales es que no es un formato *XML*, aunque se puede mapear a *XML* mediante una especificación. WSML es una familia de lenguajes, que se pueden dividir en las siguientes cinco categorías:

*WMSL-Core***:** Se encuentra en la intersección entre la lógica descriptiva y la lógica de *Horn*. La semántica se define de forma similar a *OWL-Lite*, y tiene una sintaxis basada en marcos. Es la más sencilla.

*WMSL-Flight***:** Basada en *OWL-Flight*. Además de las capacidades de *WMSL-Core*, añade restricciones y características no monótonas. Es el lenguaje preferido para modelar ontologías.

*WMSL-Rule***:** Extiende *WSL-Flight* y se basa en la programación lógica. Es el preferido para modelar objetivos y servicios *web*.

*WMSL-DL***:** Extiende *WSML-Core* con lógica descriptiva.

*WSML-Full***:** Unifica todas las variantes de WMSL bajo lógica de primer orden con extensiones no monótonas, lo que permite capturar la negación no monótona de *WMSL-Rule*.

WSMX

Es una arquitectura orientada a servicios (*SOA*) basada en eventos, que sirve como entorno de ejecución para servicios *web* semánticos. Es la implementación de referencia para *WSMO*. Se encuentra en desarrollo y está publicada como software libre en *Sourceforge6*.

- **Modelo SA*WSDL*.**

Es una extensión, desarrollada por un grupo de trabajo del *W3C*, del lenguaje de descripción de servicios *web* (*WSDL*). Está constituido por dos tipos básicos de anotaciones: la referencia al modelo y el mapeo del esquema.

Las anotaciones de referencia al modelo son las mismas que las empleadas en el modelo *WSDL*-S, se utilizan para asociar interfaces, puertos, operaciones, entradas, salidas, elementos *xml* y atributos con conceptos semánticos.

En cuanto al mapeo del esquema, son atributos añadidos a la declaración del esquema *XML* de los elementos, para especificar mapeos entre la información semántica y *XML*. Se utilizan durante la

invocación de los servicios, para formatear la información del cliente a *XML*, de forma que pueda ser entendida por el servicio *Web*, solucionándose el problema de la estructura de las entradas y salidas del servicio. Normalmente suele ir asociado a una hoja de transformación *XSLT*.

A diferencia de *WSDL*-S, permite especificar semánticamente el comportamiento de los servicios *web*, mediante la utilización de lenguajes de descripción de ontologías como *OWL*, siendo útil para la realización de coreografías de servicios. Permite el desarrollo de funcionalidades como la clasificación de servicios, el descubrimiento, el *matching*, la *c*omposición y la invocación dinámica.

1.6 Conclusiones parciales.

Con la aparición de la *web* semántica y el auge de las arquitecturas *SOA* aparecen los servicios *web* semánticos como tecnología de alternativa. En la *web* extendida, la información se encuentra bien definida, de forma que puede ser interpretada por las máquinas, permitiendo la automatización de tareas que en la actualidad solamente pueden realizarse en tiempo de compilación, como son la composición, el descubrimiento dinámico o la ejecución de servicios.

Actualmente existen diversos proyectos sobre *SWS* terminados y otros en fase de desarrollo, sin embargo, al ser una tecnología poco aplicada, el avance en este sentido es lento. Hay mucha formalización, pero pocas plataformas capaces de ofrecer todas las funcionalidades requeridas por los servicios *web* semánticos.

Los lenguajes de ontología como *OWL-S* brindan una herramienta para poder incorporar características semánticas a los sistemas facilitando el descubrimiento y utilización de servicios en forma dinámica.

Capítulo 2: Los procesos de gestación y aprobación de programas postgrado y conocimientos asociados.

Los procedimientos del proceso de Maestrías y Especialidades consisten en un conjunto estructurado de tareas o actividades estrechamente relacionadas entre sí, y el objetivo que persiguen es formar másteres y especialistas. La organización donde tienen lugar estas actividades o tareas es el ámbito de la educación de postgrado en las instituciones de educación superior y entidades científicas autorizadas y calificadas para ello, con sus canales correspondientes tales como: facultades y sus departamentos docentes y centros de estudio o investigación, la dirección de postgrado, la secretaría general y las secretarías de las áreas, entre otros.

La gestación y aprobación de programas de postgrado se considera definitivamente un proceso, pues consta de varias etapas para alcanzar su fin deseable. Dichas etapas pueden también analizarse como otros procesos bien estructurados y definidos que en conjunto y consecuentemente persiguen el objetivo final dentro del ambiente organizacional.

2.1 Proceso de gestación y aprobación de programas de maestrías y especialidades.

La gestación y aprobación de nuevos programas de maestrías y especialidades puede darse de dos maneras: se puede crear un nuevo programa de postgrado académico (PGA) o el Comité Académico (CA) puede trabajar en la redacción de un programa a partir de otro ya aprobado por la Comisión Asesora para la Educación de Postgrado (COPEP) del Ministerio de la Educación Superior (MES) para otro centro (MES, 2004, Moreno_Rodríguez *et al.*, 2010) .

De acuerdo a Moreno_Rodríguez *et al.* (2010), en los procedimientos para la gestación y aprobación de un PGA, el primer caso comienza con una nueva necesidad de un PGA por parte del centro de producción o de servicios, o surge como una nueva propuesta de PGA por profesores de un departamento docente. Dicha propuesta es analizada por el departamento docente, al cual le llega la solicitud. Si el PGA es una especialidad de postgrado, el departamento tiene que consultar al Organismo de Administración Central del Estado (OACE) de la entidad solicitante, el cual, en caso de aprobación, propone especialistas para el CA y el claustro.

El departamento docente encargado de procesar la solicitud propone al decano o director la composición del CA para el PGA, si no es aprobada la propuesta, la misma es retornada al departamento. Después de aprobación (puede ser en Consejo de Dirección [CD] o Consejo

Científico [CC]) el decano o director propone la composición del CA al rector, quien analiza la propuesta y en caso de ser aprobado el CA emite una resolución rectoral que certifica la autorización a ese CA. En caso de no aprobar la propuesta se retorna al decano o director correspondiente.

Una vez que ha sido aprobada la constitución del CA, este trabaja en la redacción del contenido del PGA y lo presenta al CC del área donde se analiza la solicitud, quien es el encargado de analizarlo y aprobarlo. Si el programa no es aprobado se trabaja nuevamente en su redacción para presentarlo nuevamente al CC. Si el programa es aprobado entonces el CA presenta el programa a la COPEP-UCLV, la cual la remite a una comisión *ad-hoc* y esta la analiza y presenta sus recomendaciones en la reunión mensual de ese órgano, incluyendo su aprobación. Si no es aprobado, el programa se regresa de nuevo al CA. Si es aprobado se emite un dictamen de COPEP-UCLV y la dirección de postgrado remite ese dictamen y el programa aprobado por COPEP-UCLV a la COPEP-MES, organismo que entrega a una comisión correspondiente el programa la cual lo analiza y propone o no aprobación definitiva. Si no es aprobado, el programa se regresa de nuevo al CA. Si se aprueba, la COPEP-MES emite dictamen de aprobación y remite al Ministro, el cual emite una Resolución Ministerial que permite la ejecución del PGA dado.

Para el caso en que se desee impartir un plan PGA ya aprobado por la COPEP MES para otro centro, es necesario solicitar una Carta-Aval del centro de origen que autorice al otro centro a impartir dicho programa con las adecuaciones correspondientes.

El proceso de gestación termina cuando en algunas de las etapas no se aprueban alguna de las presentaciones del CA o del PGA por las instancias que analizan las propuestas; o bien termina cuando todas las instancias aprueban las presentaciones realizadas y se ejecuta la tarea de confeccionar el expediente del PGA con toda la documentación emitida en el proceso.

2.2 Modelado del proceso de gestación y aprobación de programas de maestrías y especialidades.

Para lograr la estandarización se ha creado *BPMN* (siglas en inglés para *Business Process Modelling Notation*), notación de la creación de modelos de procesos de negocio la cual es una iniciativa de la *BPMI* (siglas en inglés para *Business Process Modelling Initiative*) cuyo objetivo es definir una notación gráfica en común que permita dar forma a los procesos de negocio. La notación *BPMN* proporciona la posibilidad, en especial, de separar la información comercial de la

técnica (elementos técnicos del sistema de información) para maximizar su traslado de una empresa a otra. (BPMI, 2006).

Desde la aparición de *BPMN*, y mucho más desde la absorción de *BPMI* por parte de la *OMG* (siglas en inglés para *Object Managment Group*), la notación *BPMN* ha tenido un éxito notable y como consecuencia de éste éxito han ido apareciendo gran cantidad de herramientas que dan soporte a esta especificación. (BPMI, 2007):

Una de las herramientas más populares lo constituye el *BizAgi*, se encarga de manejar el ciclo de vida de los procesos de negocio, como por ejemplo: automatización, ejecución y gestión y mejoramiento continuo). Con el modelador *BizAgi*, se pueden hacer diagramas y documentar los procesos de una manera más eficiente, buscando fomentar la colaboración en la organización. El software soporta todos los elementos necesarios para la modelación de un proceso. Estos elementos son bien descritos por *BPMI* (2006).

Partiendo de los procesos descritos en el epígrafe precedente, la modelación de los mismos ha sido tema en trabajos de investigación de años anteriores. Benítez_Concepción (2010) planteó la modelación desde el punto de vista de todos los procesos que intervienen en la gestión de un PGA, a saber que también se incluyen otros modalidades de postgrados como son, cursos, diplomados, entrenamientos, doctorados, así como también los procesos de matrícula, culminación y auditoría.

El proceso de gestación y aprobación de maestrías y especialidades constituye un subproceso particular de ese gran proceso de postgrados, regido por el MES.

En la siguiente figura (Figura 2.1.) se modela de forma detallada cada tarea o actividad que enrolan los procedimientos descritos por (Moreno_Rodríguez *et al.*, 2010), así como el flujo de mensajes entre los actores o responsables de las tareas que participan en el proceso de negocio. Seguidamente se describen de forma tabular estas mencionadas tareas, el tipo de tarea, además sus parámetros de entrada y de salida.

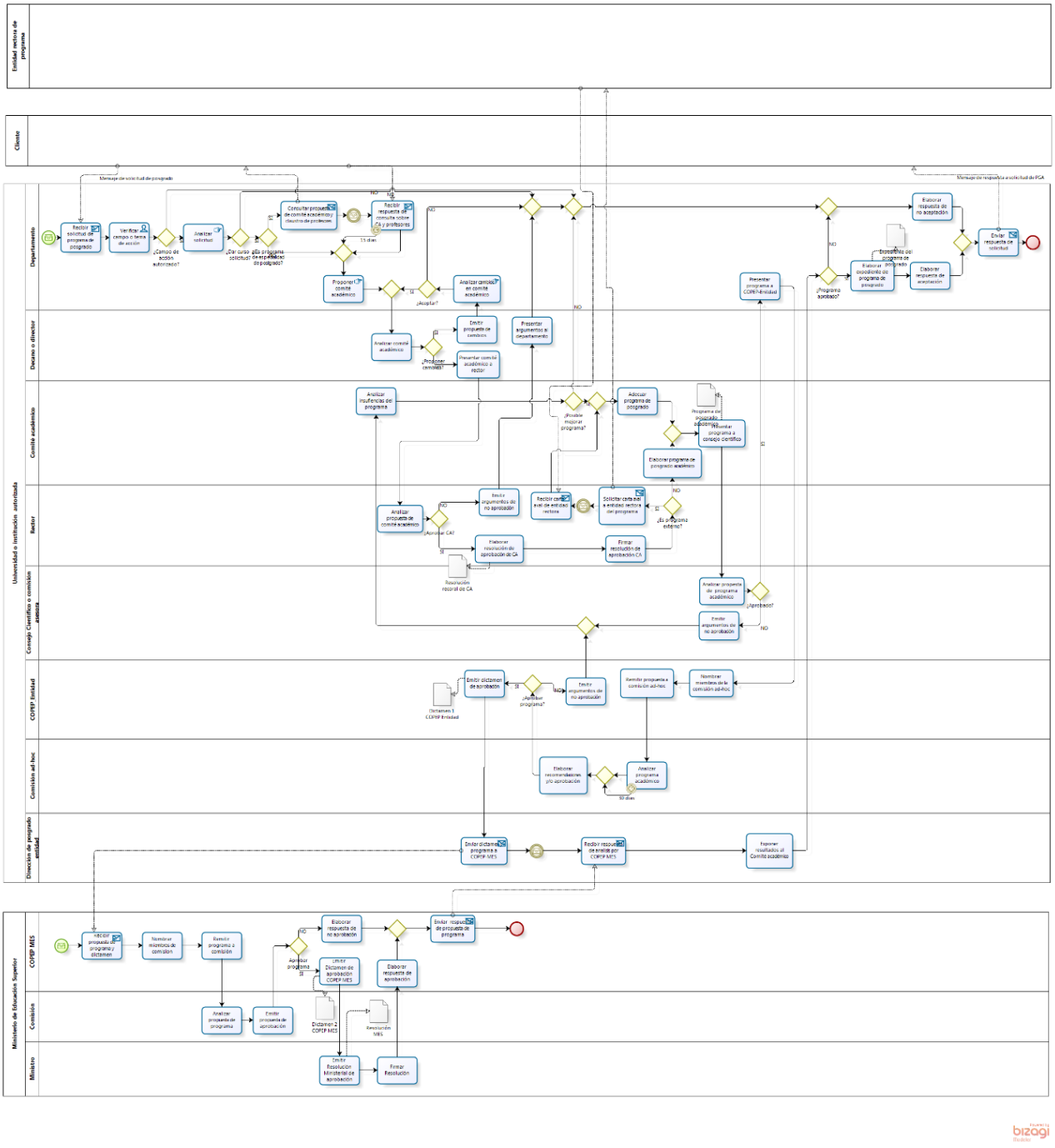


Figura 2.1: Modelado del proceso de gestación y aprobación de nuevos programas de Maestrías y Especialidades

A continuación se describen cada una de las tareas involucradas en ese modelado de proceso de negocio (Tabla 2.1).

Tabla 2.1: Descripción de las actividades del modelado de proceso.

Nombre del proceso	**Descripción**	**Tipo de Tarea**	**Entrada**	**Salida**
Recibir solicitud de programa de postgrado.	Propuesta del programa de PGA por profesores del departamento.	Recibo de mensaje.	Mensaje de solicitud o consulta.	Tema de Postgrado. Tipo PGA.
Verificar tema o campo de acción	Se verifica que el tema o campo de acción esté dentro de los posibles o autorizados por el MES	Usuario	Tema de PGA	Respuesta (si o no).
Analizar solicitud.	Analizar propuesta por el departamento correspondiente.	Manual.	Tema de Postgrado. Tipo PGA.	Respuesta (si o no). Tipo PGA.
Consultar propuesta del Comité Académico.	Si el PGA= especialidad de postgrado entonces consultar a OACE su parecer y solicitar especialistas para el CA y el claustro.	Envío de mensaje.	Mensaje de solicitud o consulta.	Mensaje.
Recibir propuesta del CA.	La OACE envía mensaje de respuesta a la solicitud de propuesta de CA y claustro.	Recibo de mensaje.	Mensaje de respuesta.	Lista de profesores.
Proponer comité académico.	Departamento propone al Decano composición del Comité Académico.	Manual.	Tema de Postgrado. Tipo PGA.	Lista de profesores.
Analizar comité académico.	Si el Decano aprueba la propuesta propone composición del CA al Rector. En caso contrario regresa la propuesta al departamento.	Manual.	Lista de profesores.	Respuesta de aprobado o posible modificación.
Analizar propuesta de Comité Académico.	Rector analiza y aprueba la composición del CA-PGA. En caso de que no lo apruebe retorna la solicitud al Decano.	Manual.	Lista de profesores.	Aprobado o no.
Elaborar resolución de aprobación de CA.	El Rector elabora y firma la resolución de aprobación de CA.	Usuario.	Lista de profesores.	Documento resolución de aprobación del CA.
Solicitar carta aval a entidad rectora del programa.	El Rector solicita la carta de aval a la entidad rectora del programa.	Envío de mensaje.	Mensaje de solicitud o consulta.	Mensaje.
Recibir carta de aval de la entidad rectora.	Se recibe el aval de la entidad que dirige el PGA.	Recibo de mensaje.	Mensaje de respuesta.	Documento aval para impartir PGA.

Elaborar programa de postgrado académico.	CA presenta Programa de PGA al Consejo Científico del área correspondiente.	Manual.	Tema de postgrado. Tipo PGA.	Documento programa de contenido del PGA.
Analizar propuesta de programa académico.	Consejo Científico analiza y aprueba. En caso de que no lo apruebe lo retorna al CA.	Manual.	Documento programa de contenido del PGA.	Aprobado o no.
Presentar programa a COPEP-UCLV.	CA presenta programa a COPEP-UCLV.	Manual.	Documento programa de contenido del PGA.	-
Nombrar miembros de la comisión ad-hoc.	COPEP-UCLV nombra y crea una comisión para analizar el PGA en cuestión.	Manual.	Tema de postgrado. Tipo PGA.	Lista de integrantes de comisión.
Analizar programa académico por comisión ad-hoc.	La comisión ad-hoc analiza y presenta en comisión mensual de aprobación. En caso de que no lo apruebe retorna la solicitud al CA.	Manual.	Documento programa de contenido del PGA.	Aprobado o no.
Emitir dictamen de aprobado.	COPEP-UCLV emite Dictamen.	Manual.	Documento programa de contenido del PGA.	Documento dictamen COPEP-UCLV de aprobación del PGA.
Enviar dictamen y programa a COPEP-MES.	Dirección de PG remite Dictamen y Programa aprobado por COPEP-UCLV a COPEP-MES.	Envío de mensaje.	Documento dictamen COPEP-UCLV y Documento PGA.	Mensaje.
Nombrar miembros de la comisión ad-hoc.	COPEP-MES nombra y crea una comisión para analizar el PGA en cuestión.	Manual.	Tema de postgrado. Tipo PGA.	Lista de integrantes de comisión.
Analizar propuesta de programa por COPEP-MES.	La comisión ad-hoc analiza y presenta en comisión mensual de aprobación. En caso de que no lo apruebe retorna la solicitud al CA.	Manual.	Documento programa de contenido del PGA.	Aprobado o no.
Emitir propuesta de aprobación	COPEP-MES emite Dictamen.	Manual.	Documento programa de contenido del PGA.	Documento dictamen COPEP-MES de aprobación del PGA.

Emitir Resolución Ministerial.	El Ministro emite y firma Resolución Ministerial que permite la ejecución del programa dado.	Usuario.	Documento dictamen COPEP-MES de aprobación del PGA.	Documento resolución ministerial COPEP-MES de aprobación del PGA.
Enviar respuesta de propuesta de programa.	COPEP-MES envía a la dirección de postgrado la respuesta de aprobación.	Envío de mensaje.	Mensaje de respuesta.	Aprobado o no.
Exponer resultados al Departamento.	Si el programa no es aprobado, se le dice al cliente.	Manual.	Resultados de las actividades anteriores.	-
Elaborar expediente del programa de postgrado.	A partir de los documentos creados se elabora un expediente del PGA para su ejecución y posteriores controles.	Usuario.	Documentos generados en los procesos anteriores.	Documento expediente del PGA.

2.3 Ontología del proceso de gestación y aprobación de programas de maestrías y especialidades.

La ontología Ontopost se creó con la finalidad de ser la base de conocimiento de un sistema semántico para la gestión de las actividades de postgrado y formalizar parte del conocimiento asociado al dominio, específicamente para los procesos de gestación y aprobación de maestría y especialidades de postgrado en la UCLV (Falcón_Espinosa, 2009).

Un aspecto a considerar dentro del trabajo es como se beneficia a Ontopost de otros formalismos de representación de conocimientos con propiedades diferentes a aquellas presentes en la misma. Particularmente, se propuso extenderla a elementos, clases o aspectos que tengan referentes al modelo ontológico *OWL-S* (Valledor_Pellicer, 2006), para la incorporación de semántica a los procesos de postgrado descritos en los epígrafes anteriores.

Se utilizó el *software Protégé* para su edición y gestión, así como para obtener los resultados correspondientes a la taxonomía, la consistencia y las clases inferidas. *Protégé*, como herramienta para la implementación de la ontología cumple con los requisitos suficientes para demostrar las ventajas del uso de la ontología como apoyo al proceso de auditoría del conocimiento.

La figura siguiente muestra el resultado de la integración o incorporación del modelo ontológico *OWL-S* a la ontología Ontopost, para el proceso de instanciación semántica de los servicios *web*.

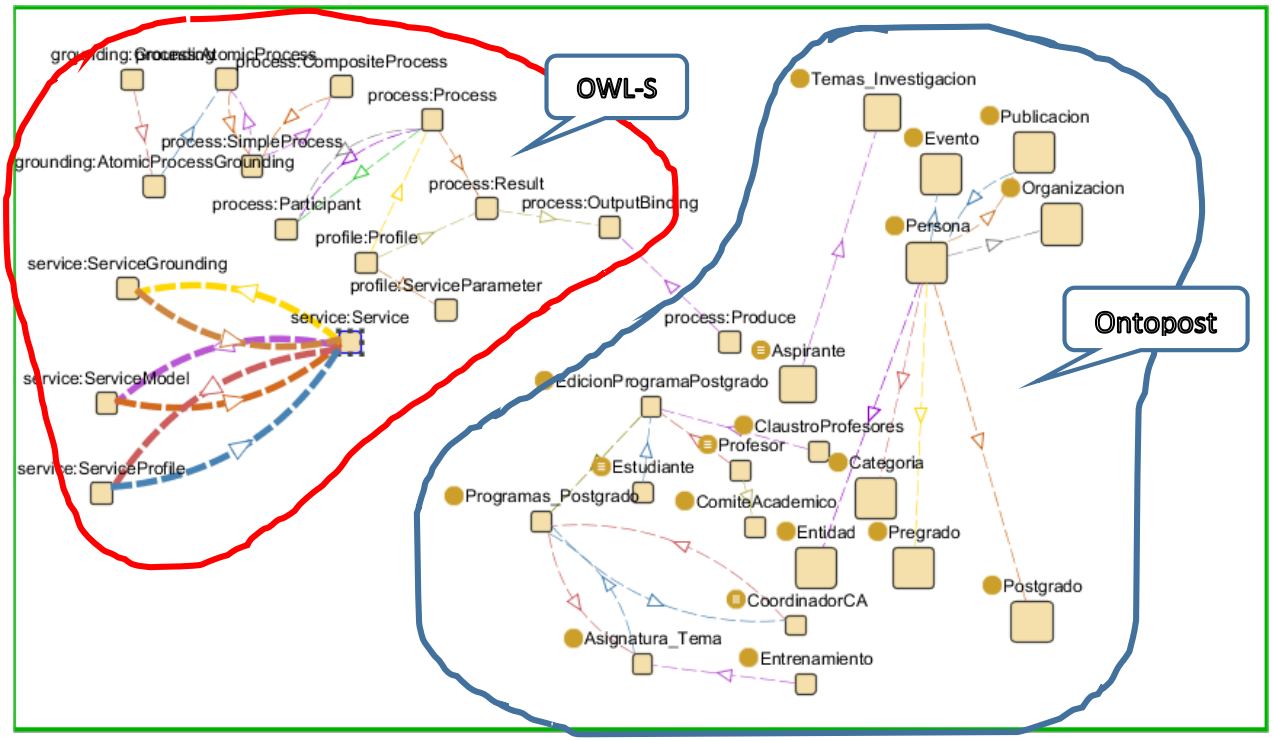


Figura 2.2: Vista parcial del resultado de la reutilización de ontologías mediante la integración o combinación de dos ontologías.

2.4 Conclusiones parciales.

La descripción semántica de los procesos y los servicios *web* y el diseño de un modelo de gestión del conocimiento que soporten los objetivos de una organización, deben dejar claro la forma en que las personas y grupos de la organización administran el conocimiento en el mundo real y a partir del cual se puedan trazar estrategias en la organización. La descripción de los modelos y el conocimiento asociado a los procesos tratados en este capítulo reflejan muy bien los conceptos y relaciones del tema tratado.

La implementación de un modelo de gestión del conocimiento de una organización involucra el compromiso de sus miembros en la creación de las condiciones tecnológicas y de toda la infraestructura para ser capaz de soportar la *web* semántica, posibilitando la búsqueda y gestión en los sistemas de consultas. Los modelos aquí representados permiten la gestión de todo el conocimiento asociado a los procesos gestación de programas de postgrado académico.

Capítulo 3: Creación de los servicios *web* semánticos para los procesos de gestación y aprobación de programas de postgrado.

3.1 Arquitectura orientada a servicio de los procesos.

Partiendo del conocimiento de los principales elementos descritos en el capítulo anterior sobre el proceso de postgrado, y en particular sobre el de gestación y aprobación de programas de postgrado, se puede definir a la estrategia ÁGIL, como principio a seguir en la definición de los servicios a implementar.

3.1.1 Análisis orientado a servicios.

En esta etapa se debe tener en cuenta los requisitos del negocio e identificar posibles sistemas automatizados a aplicar en el proceso, para luego decidir sobre un modelo de servicios candidatos. En la misma se definen los servicios a ser construidos y qué lógica del negocio debería ser encapsulada por cada servicio, es decir, definir las operaciones y agruparlas en un contexto lógico (servicios candidatos); definir los límites de estos servicios para no recubrir otros existentes que puedan ser reutilizados; identificar lógica encapsulada para posible reutilización y definir cualquier composición de los servicios definidos.

En el proceso de gestación y aprobación de especialidades y maestrías se hace necesario, por ejemplo, poner información a disposición de los posibles clientes una lista de temas autorizados para programas de postgrado académico, como catálogo de referencia rápida, y de esta forma poder elegir entre los posibles departamentos a impartir el tema. La mejor manera de listar o publicar los datos entre los posibles usuarios en general, situados a veces en diferentes ubicaciones, resulta ser un servicio *web*. Además debe tenerse la posibilidad, para un usuario con privilegio, de crear una instancia del proceso de gestación y aprobación para un programa académico, cuyo tema no esté aún dentro del plan de ofertas de programas autorizados. Durante la ejecución de una instancia del proceso de aprobación de un programa se hace necesario ejecutar servicios para las tareas de mensajerías entre actores y clientes del proceso; también servicios para la impresión, carga y salva de documentos y el chequeo de estado de los procesos.

Es imprescindible además, definir las siguientes tareas, las cuales serán desarrolladas en los acápites siguientes:

- Seleccionar el nombre de los servicios *web*.

- Desde donde se ejecutará el servicio (lugar de ubicación, ejemplo en un directorio del servidor interno llamado /*Program Files/Apache Software Foundation/Tomcat 6.0/webapps/axis2*).
- Elegir la manera de comunicarse los usuarios con los servicios *web*: Las aplicaciones cliente del servicio *web* enviarán y recibirán mensajes *SOAP* / con codificación *RPC*. Se podrá acceder a través de un navegador *web* por el protocolo *http*.
- Decidir las operaciones que llevarán a cabo los servicios *web*: Se seleccionaron tres operaciones: obtener la lista de ofertas de programas de postgrado académico, seleccionar un programa en particular y mostrar la información detallada sobre el postgrado en particular.
- Determinar los parámetros necesarios para cada operación, como son los parámetros de entrada y los que se devolverán como resultado.
- Dar respuesta a preguntas de competencias en el momento de consultas, por ejemplo; obtener lista de programas académicos autorizados según una categoría o tipo. Listar ofertas del tipo de programa de postgrado académico. Mostrar detalles del programa (el parámetro de entrada es el identificador único del programa y el resultado es la descripción del programa y su coordinador), etc.
- Decidir sobre los tipos de datos o estructuras de datos que exige cada uno de los parámetros de entrada y salida.

3.1.2 Diseño orientado a servicios.

Como servicios candidatos fueron definidos los siguientes:

- Servicios de envío y recibo de mensajería
  - Solicitud de aprobación de postgrado.
  - Consultar propuesta de C.A.
  - Solicitud de avales
- Servicio de crear, guardar y cargar documentos en y desde *XML*.
  - Resolución rectoral de aprobación del C.A.
  - Carta de aval de entidad rectora de PGA
  - Documento de contenido del PGA.

- Dictamen de aprobación de COPEP_UCLV
- Dictamen de aprobación de COPEP_MES
- Resolución ministerial de aprobación PGA
- Expediente PGA

➢ Servicios de consultas.

- Lista de temas.
- Lista de departamentos.
- Lista de profesores.
- Lista de ofertas de programas.

3.1.3 Capas y composición de los servicios.

La posibilidad de tener una buena idea de cómo proceder con la agrupación de las etapas del proceso es importante para la composición de los servicios. En este paso se pone de manifiesto la posible relación entre la orquestación y la capa de servicios de negocio, para identificar el potencial de las composiciones de servicios.

Cualquier capa de servicio que se establezca es preliminar, y todavía sujeta a revisiones durante el proceso de diseño. En este caso se decide aplicar la configuración mostrada en la figura 3.1, en la cual se puede apreciar una propuesta de cómo agrupar las composiciones de servicios candidatos de las distintas etapas del proceso. La aparición de servicios nuevos que surgen de la necesidad de lograr una mejor composición, es una característica remarcable de esta arquitectura. Estos servicios se encargan del control de las operaciones que se relacionan con la lógica de los subprocesos y actúan como servicios controladores de las composiciones de servicios.

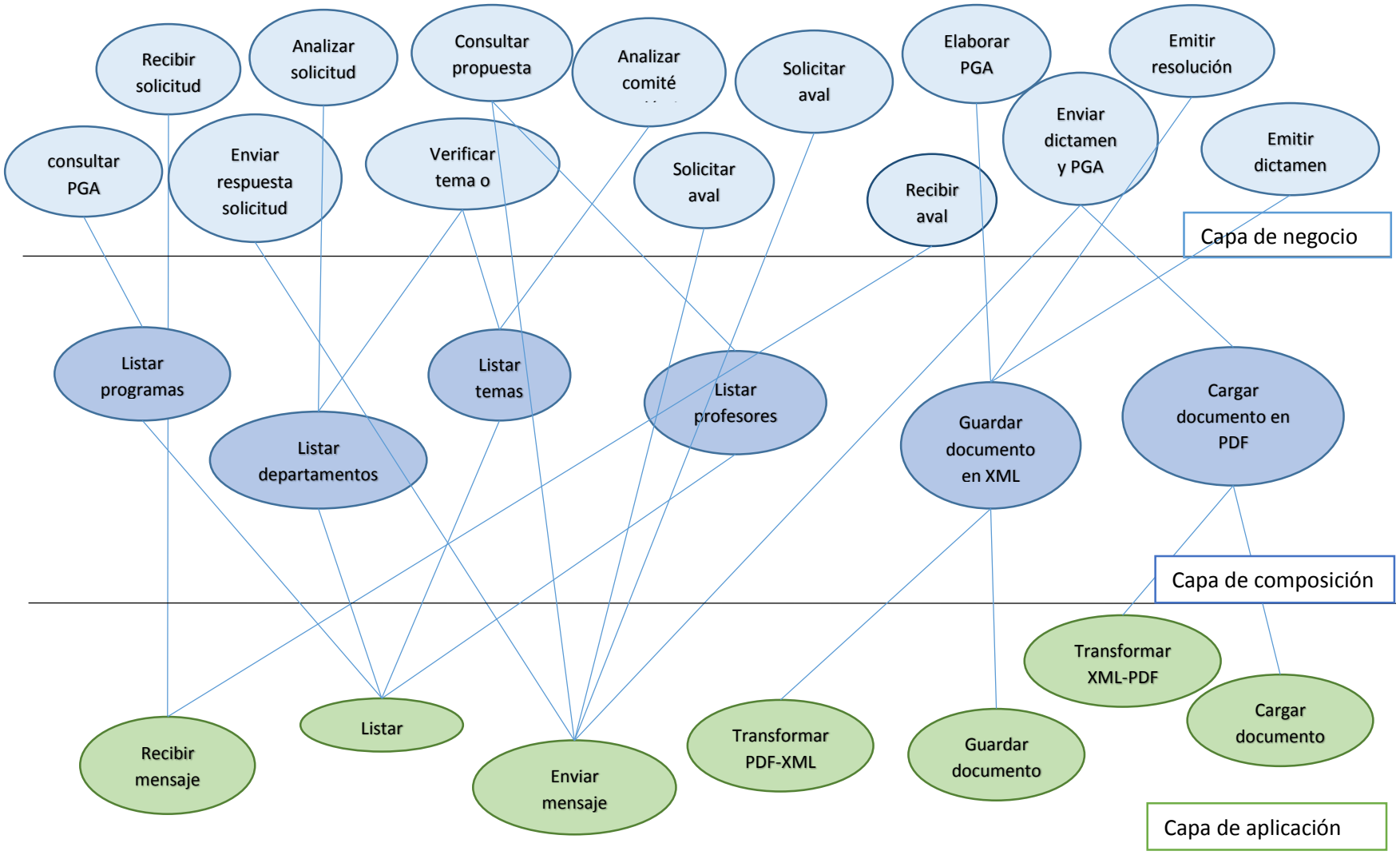


Figura 3.1: Capas de servicios aplicadas al proceso de gestación y aprobación de programas de postgrado.

3.2 Implementación de los servicios *web*.

3.2.1 Creación o exposición.

Para la creación de los servicios se utilizó el software Eclipse Juno. Este software ofrece la posibilidad de desarrollar servicios web mediante dos enfoques.

Una de las estrategias de desarrollo de servicios web es el *bottom up* (de abajo hacia arriba), en la cual se parte de la codificación de la lógica del negocio en lenguaje *Java*, para luego crear el archivo *.aar*, donde se constituye la composición del servicio, y de esta manera obtener el archivo *.wsdl*, descriptor del servicio, a partir de su publicación en un servidor repositorio de servicios web. El archivo *.wsdl*, puede ser consumido y también ser modificado o mejorado mediante el software *XMLSpy*. La segunda estrategia es el *top down* (de arriba hacia abajo), que ofrece la opción de construir el servicio web a partir de un archivo *.wsdl* importado, el cual junto a la especificación java conforman la composición del servicio web deseado (Eclipse, 2003).

La estrategia aplicada para la implementación de los servicios web de la presente investigación fue la "*bottom up"*, ya que fue necesario crear primeramente en *Java* las clases y las funciones que describen la lógica del negocio, para ganar en claridad en este sentido. Por ejemplo: para codificar el servicio "SolicitudAprobPGA" fue necesario programar dos funciones o servicios atómicos más; por un lado se debe listar los temas autorizados y por otro los departamentos que están facultados para impartir esos temas, con el objetivo de verificar los temas autorizados y los departamentos que ofrecen estos temas. Estos dos servicios atómicos establecen una secuencia lógica para conformar el servicio de SolicitudAprobPGA, con el objetivo de conocer si el tema propuesto está disponible y si existe el departamento que ofrece la impartición. En la siguiente figura (Figura 3.2) se muestra un ejemplo de código *Java* de esta especificación.

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

          String []comision= {"Departamento", "CA_Decano","CA_Rector", "PGA_CC","PGA_COPEP" };
          boolean []aprob= {false, false,false, false,false };
          String str="";

           System.out.println("Tema: "+ Seleccionartema());
           System.out.println("Departamento: "+ SelecionarDpto());

           for (int i = 0; i < comision.length; i++) {

               str=System.console().readLine("Desea aprobar el tema(responda si o no) "+comision[i]);
               if(str.equals("si")){
                   aprob[i]=true;
               }else
                   aprob[i]=false;
           }
           if (aprob[0]==true&&aprob[1]==true&&aprob[2]==true&&aprob[3]==true) {
               System.out.println("Tema Autorizado");
           }else
               System.out.println("Tema Desaprobado");

              }
```

Figura 3.2: Código *Java* del servicio Solicitud de Aprobación de Postgrado

En el proceso de creación del servicio web, se pudo apreciar que el software eclipse tiene también la opción de crear el fichero descriptor del servicio *.wsdl*. Como resultado de la edición y gestión del servicio, se crea una estructura de archivos y carpetas, la cual se muestra en el panel izquierdo, correspondiente al *Explorer* del *Windows* de la figura 3.3.

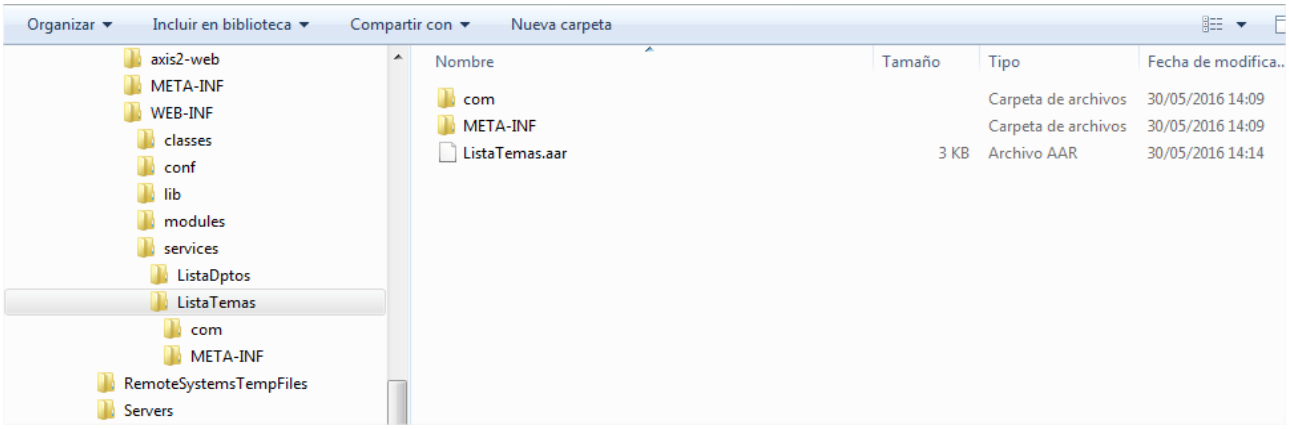


Figura 3.3: Estructura de carpetas creada en el eclipse, lista para crear la composición *.aar*

Seguidamente, el *Java* posee entre sus opciones, la posibilidad de componer toda esta estructura en un solo archivo, cuyo formato debe ser reconocido por el servidor de aplicaciones, que en este

caso resulta ser *Apache-Axis2*. La siguiente figura (Figura 3.4) muestra la secuencia de comandos para la composición de los archivos creados en el software en la creación de servicios *web*.

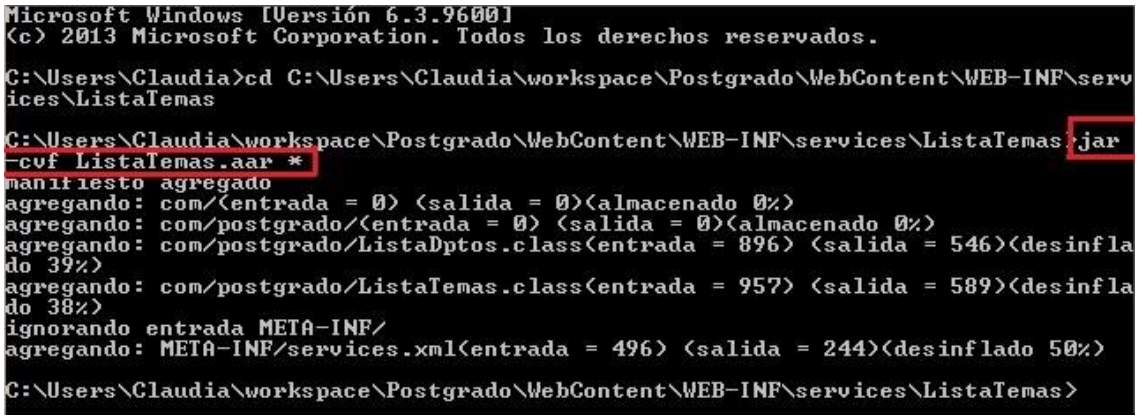


Figura 3.4: Secuencia de instrucciones para la composición de un servicio web.

3.2.2 Disposición o publicación.

Para esta etapa de publicación de los servicios *web* implementados se hace necesario crear las condiciones de instalación de los servidores que gestionarán la publicación y el consumo de los servicios. Partiendo de la correcta instalación del servidor *Apache TomCat* y de *Axis2* como un servidor de servicios web, desplegado en *Apache* se debe guardar en la carpeta de servicios del *Axis2*. La figura 3.5 muestra el resultado de los servicios web publicados y listos para el consumo, cuyas descripciones *.wsdl* pueden ser accedidas a partir de un clic en el nombre del mismo.

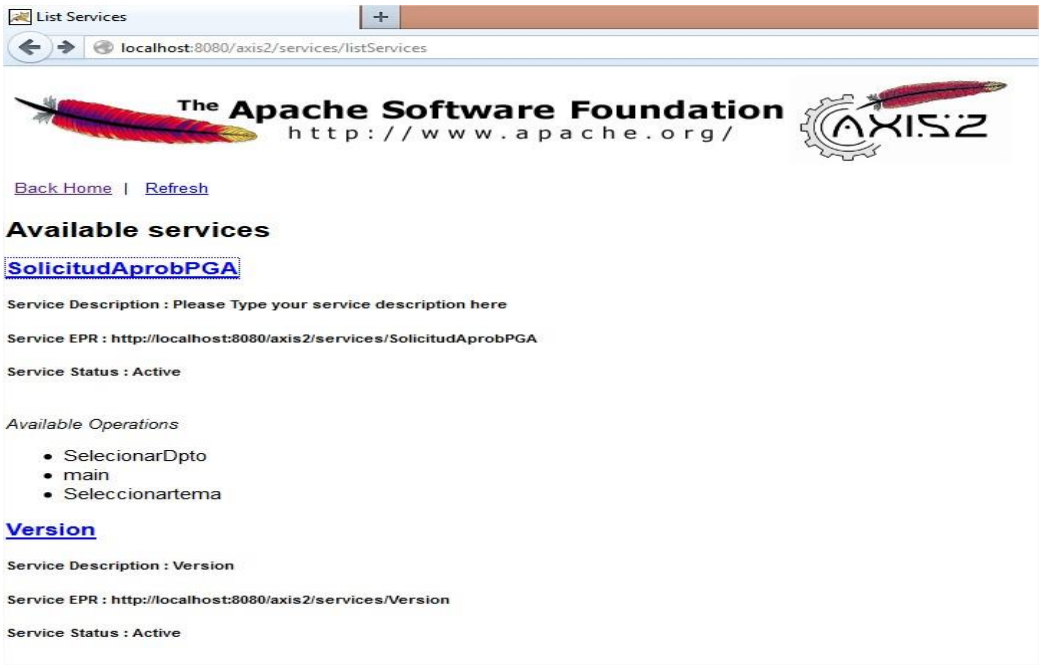


Figura 3.5: Servicio *web* publicado dentro del repositorio de *Axis2.*

En la figura 3.6 se puede observar el esquema de la estructura del archivo *.wsdl* del servicio *web* implementado. Se muestran las definiciones que componen la descripción del servicio, los parámetros de entrada, los parámetros de salidas, los respectivos mensajes, etc. La implementación de los servicios puede ser referenciada en el anexo 1 de este documento.

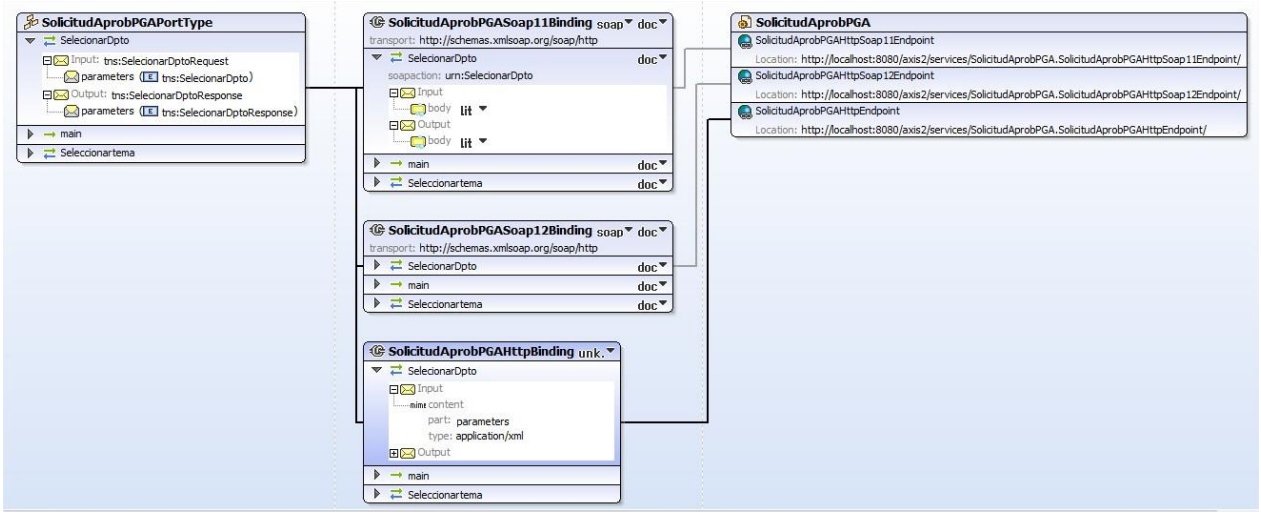


Figura 3.6: Esquema *WSDL* del servicio Solicitud de Aprobación de Postgrado.

3.3 Incorporación semántica a los servicios web. Implementación de los *SWS* con *OWL-S*.

La propuesta se basa en agregar una nueva dimensión al modelo de proceso que expone el perfil. De esta forma se agrega un nuevo tipo información a la descripción funcional que muestra los posibles servicios. La estructura

Clase Servicio Perfil (*ServiceProfile*)

La clase de servicio perfil proporciona una superclase de todos los tipos de descripción de alto nivel del servicio. No impone ninguna representación de los servicios, sino que ordena la información básica para vincular cualquier instancia del perfil con una instancia del servicio.

Existe una relación bidireccional entre un servicio y un perfil, de manera que un servicio puede estar relacionado con un perfil y un perfil para un servicio. Estas relaciones se expresan mediante las propiedades de objeto *presents* y *presentedBy*.

Algunas propiedades del perfil proporcionan información legible por humanos que es poco probable que sea procesada de forma automática. Estas propiedades incluyen *serviceName*, *textdescription* y *contactInformation*. Un perfil puede tener como máximo un nombre de servicio y descripción de texto, pero la mayor cantidad de elementos de información de contacto como el proveedor quiere ofrecer. En el anexo 3 se muestra la interface de instanciación que brinda.

Funcionalidad: El perfil *OWL-S* representa dos aspectos de la funcionalidad del servicio: la transformación de la información (representado por las entradas y salidas) y el cambio de estado producido por la ejecución del servicio (representados por las precondiciones y efectos).

La clase perfil de la ontología define las siguientes propiedades de los apuntadores a las entradas y salidas: *hasParameter, hasInput, hasOutput, hasPrecondition, hasResult*

En la siguiente figura, se presenta el esquema estructural de clases e instancias necesitadas como parámetros de la información para la creación de la instancia de la clase *Profile*, Este conjunto de información será expuesto en el registro de servicios (Figura 3.7).

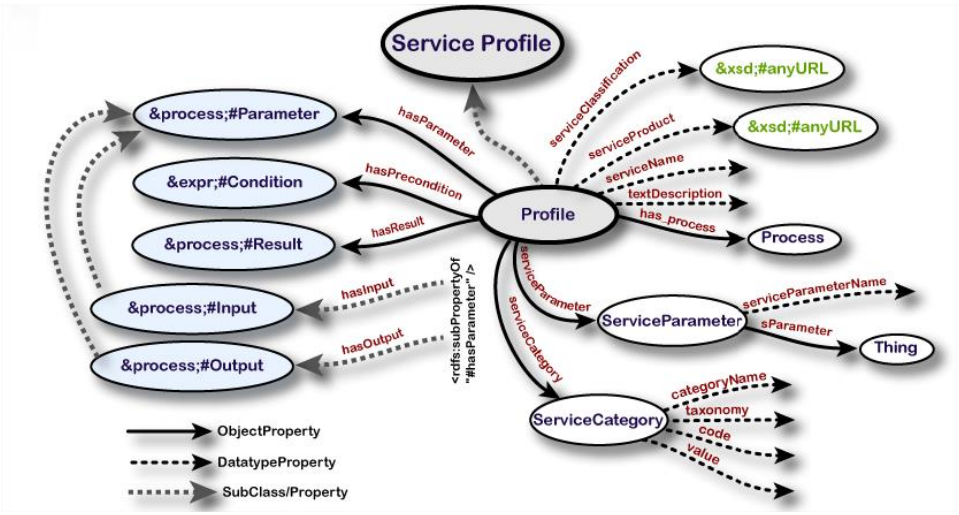


Figura 3.7: Relación de clases, parámetros e instancias de la clase *Service Profile.*

Servicio de modelado de proceso (*Service Model* o *Process*)

Para dar una perspectiva detallada sobre cómo interactuar con un servicio, este puede verse como un proceso. En concreto, *OWL-S* define una subclase de *ServiceModel o Process*, que se basa en el trabajo bien establecido en una variedad de campos de las ciencias de la computación (Martin *et al.*, 2004).

Las entradas y salidas son subclases de una clase general llamada de parámetros. Es muy práctico para identificar los parámetros con lo que se llaman variables en *SWRL* (*Semantic web Rules Language*), el lenguaje para expresar reglas *OWL*. Cada parámetro tiene un tipo, se especifica mediante un *URI* (*Uniform Resource Identifier*). Esta no es la clase *OWL* el parámetro pertenece a, sino una especificación de la clase (o tipo de datos) a la que los valores del parámetro pertenecen (Las entradas y salidas son subclases de parámetro). En la figura 3.8 se muestra un esquema de todos las subclases y parámetros necesitados en la especificación del servicio de modelado o proceso.

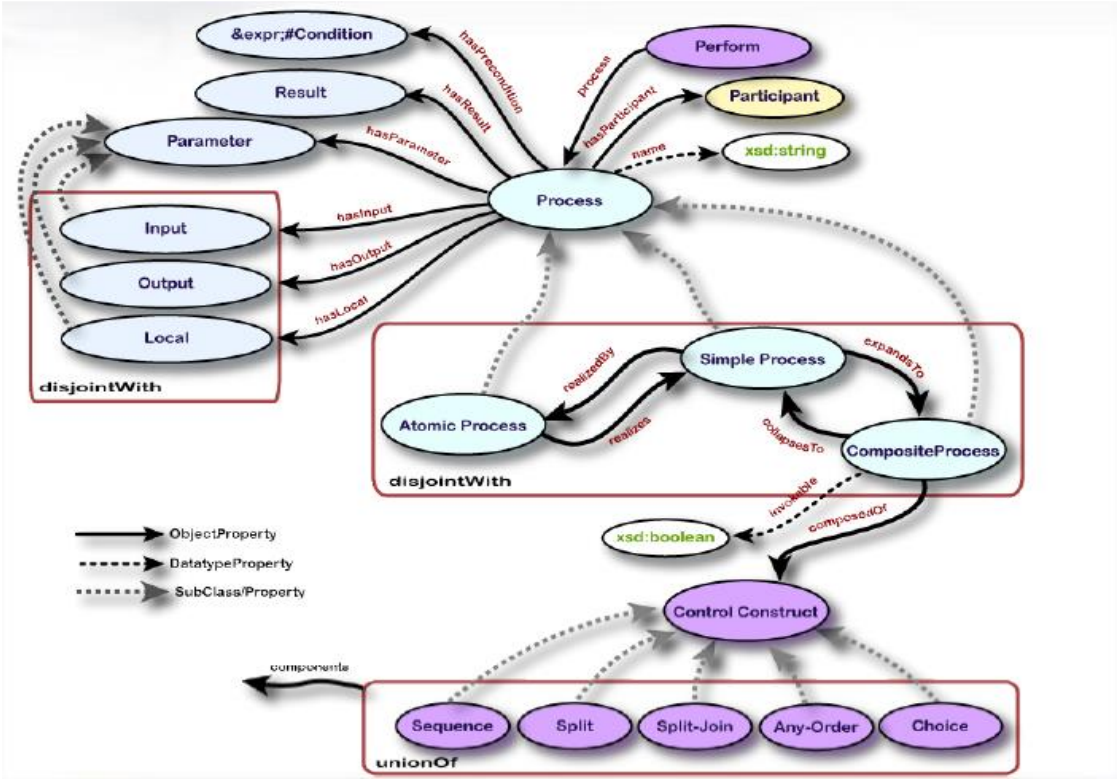


Figura 3.8: Relación de clases, parámetros e instancias de la clase *ServiceModel*.

Aquí se conectan las entidades de entrada y salida a las respectivas propiedades mostradas a continuación:

Propiedad	***Distancia***	***Tipo***
hasParticipant	Partícipe	Clase
hasInput	Entrada	Parámetro
hasOutput	Salida	Parámetro
hasLocal	Local	Parámetro
hasPrecondition	Condición	Expresión
hasResult	Resultado	Parámetro

En esta fase de implementación se formalizan las clases de procesos: atómico, simple y compuesto. Los procesos atómicos corresponden a las acciones que un servicio puede llevar a cabo mediante la participación en una sola interacción. Los procesos simples proporcionan un mecanismo de abstracción para proporcionar múltiples vistas del mismo proceso. Finalmente, los procesos

compuestos corresponden a acciones que requieren protocolos de múltiples pasos y/o acciones del servidor.

Estos mismos parámetros pueden ser especificados a partir de la herramienta o *plug-in* disponible para Protégé, denominada *OWLS*-Tab (Figura 3.9). Partiendo, por ejemplo que las relaciones entre este modelo ontológico y la ontología Ontopost se concretan por medio de los parámetros de entrada del primer modelo.

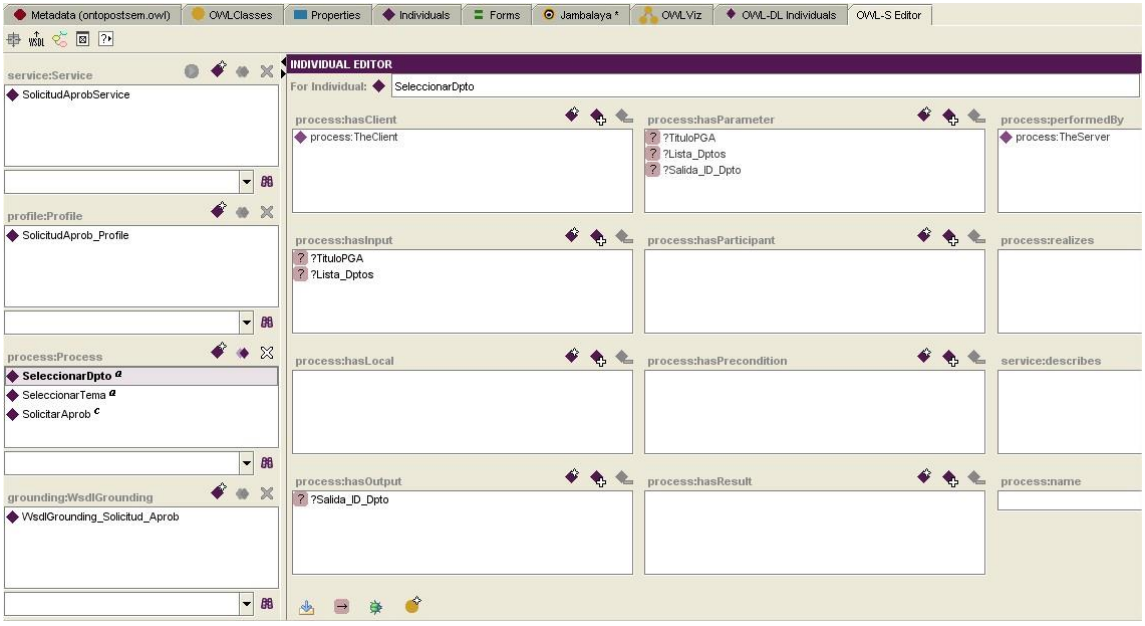


Figura 3.9: Instanciación de los procesos correspondiente a la clase *ServiceModel*

SeleccionarTema es un proceso atómico tiene como datos de entrada el *TipoPGA* y *Lista_Temas* y como dato de salida *Salida_ID_Tema*, que a su vez son los parámetros del proceso. *SeleccionarDpto* es un proceso atómico que tiene como datos de entrada el *TituloPGA* y *Lista_Dptos* y como dato de salida *Salida_ID_PGA*, los cuales son los parámetros de dicho proceso.

Para la implementación de un proceso compuesto es necesario tener en cuenta una característica crucial, esto es; un proceso compuesto es su especificación de cómo sus entradas son aceptadas por subprocesos particulares, y cómo sus diversas salidas son producidas por subprocesos particulares. En la figura 3.10 se muestra la implementación de la instanciación y entrada de parámetros para un proceso compuesto a partir de la composición de dos procesos atómicos.

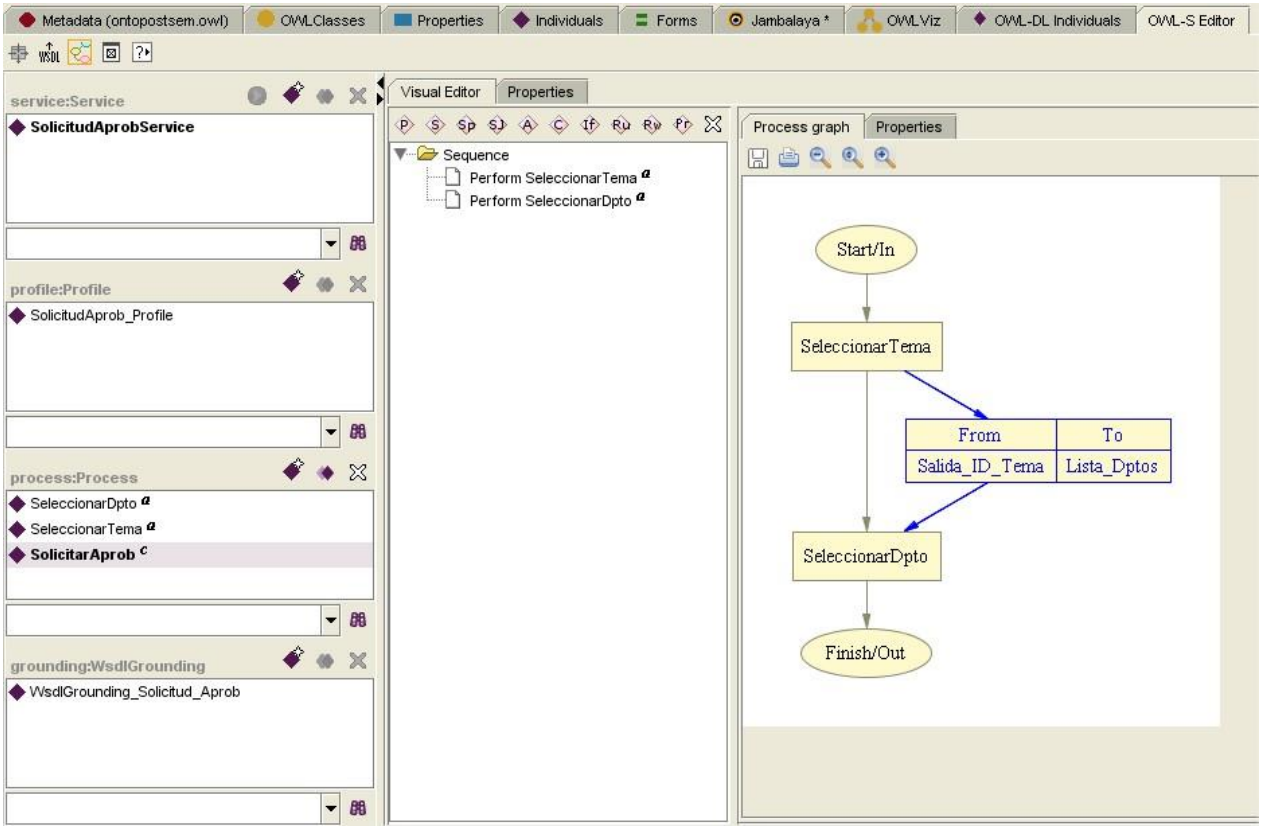


Figura 3.10: Composición del proceso SolicitarAprob a partir de los proceso atómicos *SeleccionarTema* y *SeleccionarDpto.*

Servicio de acceso (*ServiceGrounding*)

La función central de servicio de acceso del *OWL-S* es mostrar cómo las entradas y salidas de un proceso atómico han de ser descritas concretamente como mensajes, que llevan esas entradas y salidas en algún específico formato transmisible (Figura 3.11). Una puesta a tierra puede ser pensada como una asignación de un extracto de una especificación concreta de los elementos de descripción de servicios que se requieren para interactuar con el servicio - en particular, para nuestros propósitos, las entradas y salidas de los procesos atómicos. Cabe destacar que en *OWL-S*, tanto la clase *ServiceProfile* y la *ServiceModel* son consideradas como representaciones abstractas; sólo la clase *ServiceGrounding* con el nivel concreto de la especificación.

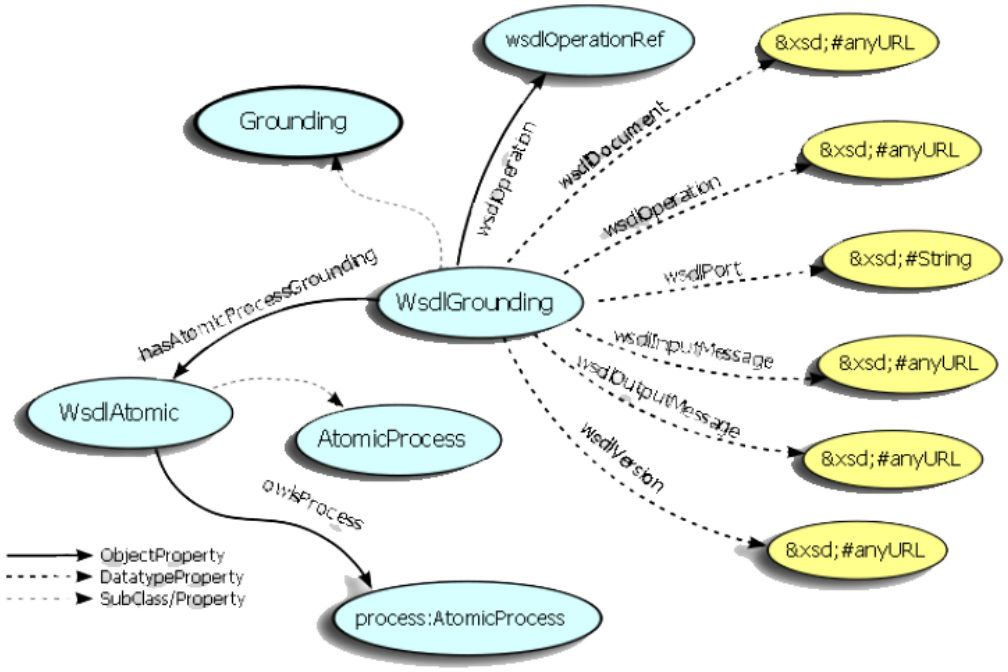


Figura 3.11: Clase *ServiceGrounding* en la ontología *OWL-S*

Luego, se realiza implementación de la clase *ServiceGrounding*, en la cual se especifica la dirección del archivo *.wsdl*, lo que constituye la importación del mismo a la ontología de servicios semánticos. La ontología resultante de esta implementación está basada en las instanciaciones anteriormente descritas, donde cada uno de los parámetros son los enlaces de las instancias de los procesos atómicos y con estos la formación del proceso compuesto (Figura 3.12).

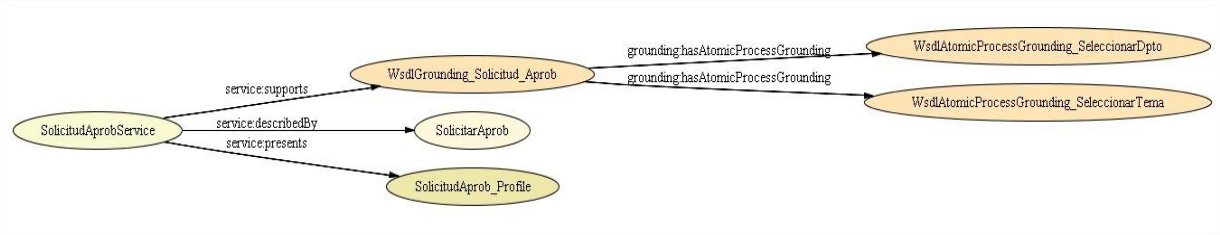


Figura 3.12: Instanciación de la ontología *OWL-S* para el proceso SolicitudPGA

3.4 Conclusiones parciales.

En un nivel superior, para el desarrollo de la *web* semántica se hace necesario la utilización de servicios *web*, los cuales posibilitan la creación de una *SOA*, en la cual lo que importa son las funcionalidades que a través de la interface de acceso del servicio web que se expone a los clientes.

Las características más interesantes de la *web* semántica son su flexibilidad y la facilidad que ofrece para añadir información y relacionarla con otra ya existente; para ello se hace necesario la utilización de ontologías que permitan añadir significado.

El estándar para especificar ontologías del *W3C* es *OWL* y está fundamentado en las lógicas descriptivas. *OWL-S* proporciona una serie de construcciones como la secuencia o la separación que permiten el control de flujo entre procesos.

Conclusiones

- Para el diseño y definición de los servicios *web* fue seleccionada la estrategia Ágil, debido a la flexibilidad de sus procedimientos.
- Fueron implementados un total de 7 servicios atómicos en la capa de aplicación, los cuales son utilizados y reutilizados por los 6 servicios definidos en la capa de composición y los 14 definidos en la capa de negocio.
- Para la incorporación de la semántica se aplicó el modelo *OWL-S*, por su factibilidad de implementación y acorde a los estándares utilizados en los procesos de postgrado.
- *Protégé* resultó ser la herramienta idónea para la incorporación de la semántica, mientras que *Eclipse* y *AltovaXML* lo fueron para la implementación *Java* y *WSDL* respectivamente. La publicación de estos servicios se logró mediante el repositorio de servicios *Axis2* desplegado en el servidor de aplicaciones *ApacheTomCat*. Todos perteneciente a la categoría de software libre
- En la incorporación de los descriptores semánticos mediante *OWL-S*, fueron creadas las instancias para los respectivos procesos de la actividad de postgrado, cuyos parámetros son definidos en la ontología Ontopost.

Recomendaciones

- Aplicar este procedimiento de diseño e implementación a los procesos sustantivos de postgrado, teniendo en cuenta la reutilización de los aquí desarrollados.
- Crear agentes para el descubrimiento de los servicios.

Bibliografía

ALTOVA*XML*. 2016. *Soluciones para servicios web* [Online]. Available: www.altova.com/es/*xml*spy.html [Accessed Abril 2016].

ASF. 2016. *Axis2 as the web service engine* [Online]. EEUU: Apache Foundation. Available: http://www.apache.org [Accessed Mayo 2016].

BENÍTEZ_CONCEPCIÓN, L. M. 2010. *Modelación de los Procesos de Negocios para la Formación de Másteres y Especialistas.* Licenciado Tesis de grado, Universidad Central "Marta Abreu" de Las Villas.

BERNERS-LEE'S, T. 2004. Tim Berners-Lee's Semantic Web. *In:* UNIVERSITY, R. A. (ed.). South Africa.

BMIR. 2016. *Protégé, a free open-source ontology editor and framework for building intelligents ssystem* [Online]. EEUU: Stanford University. Available: http://protege.stanford.edu [Accessed Mayo 2016].

BPMI, B. P. M. I. 2006. *BPMI. Business Process Notation Specification.*

BPMI, B. P. M. I. 2007. *Bpmn implementors and quotes.*

CABRAL, L., DOMINGUE, J., MOTTA, E., PAYNE, T. & HAKIMPOUR, F. 2004. Approaches to Semantic Web *Services*: An Overview and Comparisons Available: http://technologies.kmi.open.ac.uk/irs/cabralESWS04.pdf.

CALIUSCO, M. L. 2011. La Web Semántica: tecnologías y aplicaciones. *La web semántica.* Santa Fe: Universidad Técnica Nacional de Argentina, Facultad Regional de Santa Fe.

ECLIPSE, F. 2003. Eclipse platform. Technical overwie.

ERL, T. Service Oriented Architecture: Concepts, Technology, and Design. 2005 New Jersey, USA. Prentice Hall.

FALCÓN_ESPINOSA, D. 2009. *Desarrollo de una Ontología de Dominio y Modelado de procesos para el Negocio de Postgrado en la UCLV.* Tesis de Diploma Trabajo de diploma, Universidad Central "Marta Abreu" de Las Villas.

FERNÁNDEZ, B. G. 2013. *Modelación del Proceso de Negocio de Postgrado en la UCLV.* Licenciado Tesis de grado, Universidad Central "Marta Abreu" de Las Villas

FERNÁNDEZ_BREIS, J. T. 2008. *Un entorno de integración de ontologías para el desarrollo de sistemas de gestión de conocimiento.* Tesis de doctorado Doctoral, Universidad de Murcia.

GRUBER, T. R. 1993. A translation approach to portable ontology specifications. *Knowledge Acquisition.* London: UK Academic Press.

KRISHNAMURTHI, S. & BULTAN, T. 2005. Characteristics of web *services* and their impact on testing, analysis and verification. *ACM SIGSOFT, Software Engineering Notes,* Vol. 30**,** 2.

MARTIN, D., BURSTEIN, M. & HOBBS, J. 2004. *OWL-S: Semantic Markup for Web Services* [Online]. EEUU: W3C. Available: https://www.w3.org/Submission/OWL-S/.

MES 2004. Resolución Nro. 132/2004. In: SUPERIOR, M. D. E. (ed.) 132. La Habana: MES.

MICHÁN, L. 2011. Ontologías para Web 3.0 y minería de literatura. *Biiiogeek* [Online]. Available from: http://biiiogeek.blogspot.com/ [Accessed Junio Acc. 2014].

MORA_CUELLAR, O. 2010. *Modelado y Diseño de Servicios en una Arquitectura Orientada a Servicios para un Sistema de Control de la Formación de Másteres y Especialistas de Postgrado.* Licenciado Tesis de grado, Universidad Central "Marta Abreu" de Las Villas.

MORENO_ORTIZ, A. 2005. Ontologías para la Terminología: Por qué, Cuándo, Cómo. *Tradumática* [Online], V. 2014.

MORENO_RODRÍGUEZ, R., ESTRELLA PAZ_ MARTÍNEZ , PADRÓN_SOROA, S., DOMÍNGUEZ_HERNÁNDEZ, J. & MORALES_ESPINOSA, A. 2010. Procedimientos del Proceso Sustantivo de Formación de Másteres y Especialistas de Postgrado. *In:* VILLAS, U. C. M. A. D. L. (ed.). Santa Clara.

MS, M. C. 2006. La Arquitectura Orientada a Servicios(SOA) de Microsoft aplicado al mundo real. EEUU.

ORACLE 2008. Gestión de Procesos de Negocio, Arquitectura Orientada a Servicios y Web 2.0: ¿Transformación de Negocios o Problemática Global? *In:* CORP., O. (ed.) *Informe Ejecutivo de Oracle.* California, E.E.U.U.: Oracle Corporation.

ORACLE, C. 2016. *NetBeans IDE- The smarter and faster way to code* [Online]. EEUU: ORACLE Corp. Available: http://netbeans.org/features/index.html [Accessed Mayo 2016].

RAMOLLARI, E., DRANIDIS, D. & SIMONS, A. 2007. A Survey of Service Oriented Development Methodologies. *The 2nd European Young Researchers Workshop on Service Oriented Computing.* University of Leicester, England: University of Leicester.

ROCHE_BENITEZ, A. 2015. *Servicios web para la gestión de información del Departamento de Computación.* Ingeniero Tesis de grado, Universidad Central "Marta Abreu" de Las Villas.

ROMÁ_FERRY, M. T. 2009. *OntoFIS: Tecnología ontológica en el dominio farmacoterapéutico.* Tesis de doctorado Doctoral, Universidad de Alicante.

SAMPER_ ZAPATER, J. J. 2005. *Ontologías para servicios web semánticos de información de tráfico: Descripción y Herramientas de explotación.* Master Tesis de Doctorado, Universidad de Valencia.

SAN_SEGUNDO, R. & MARTÍNEZ_AVILA, M. 2012. El orden de los saberes y la organización digital. *20 años del Capítulo español de ISKO, Actas del XI Congreso ISKO (Ferrol 2011).* . Coruña, España: Universidad de Coruña.

SARWAR_BAJWA, I., KAZMI, R., MUMTAZ, S., CHOUDHARY, M. & NAWEED, M. S. SOA and BPM Partnership: A paradigm for Dynamic and Flexible Process and I.T. Management. *In:* PWASET, ed. PROCEEDINGS OF WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY., 2008.

SOLLAZZO, T., HANDSCHUH, S., STAAB, S. & FRANK, M. 2002. Semantic Web Service Architecture — Evolving Web Service Standards toward the Semantic Web. Available: http://userpages.uni-koblenz.de/~staab/Research/Publications/sub-flairs2002.pdf.

STOLLBERG, M., CRISTINA FEIER, ROMAN, D. & FENSEL, D. 2006. Semantic Web *Services* - Concepts and Technology. Available: http://www.michael-stollberg.de/publications/wsmo-bookchapter.pdf.

SYCARA, K. & MARTIN, D. 2006. Tools and Technologies for Semantic Web *Services*: Semantic Web *Services*: An OWL An OWL-S Perspective.

VALLEDOR_PELLICER, P. 2006. Servicios Web Semánticos. *In:* INFORMÁTICA, D. D. (ed.) *Cursos de doctorado.* España: Universidad de Oviedo.

VARGA, L. & HAJNAL, A. 2005. Semantic Web *Services* Description Based on Web *Services* Description. *In:* W3C (ed.) *Workshop on Frameworks for Semantics in Web Services.* EEUU: W3C.

W3C. 2004. *Web Services Glossary* [Online]. World Wide Web Consortium. Available: https://www.w3.org/TR/ws-gloss/ [Accessed 19 Mayo 2016].

Anexos

ANEXO 1

Definición Abstracta para el servicio Solicitar Aprobación de Programa de Postgrado:

```
<wsdl:types>
  <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://postgrado">
    <xs:element name="main">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="args" nillable="true" type="xs:string"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="SelecionarDpto">
      <xs:complexType>
        <xs:sequence/></xs:complexType>
    </xs:element>
    <xs:element name="SelecionarDptoResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="return" nillable="true" type="xs:string"/>
        </xs:sequence></xs:complexType>
    </xs:element>
    <xs:element name="Seleccionartema">
      <xs:complexType>
        <xs:sequence/>
      </xs:complexType>
    </xs:element>
    <xs:element name="SeleccionartemaResponse">
      <xs:complexType>
        <xs:sequence>
          <xs:element minOccurs="0" name="return" nillable="true" type="xs:string"/>
        </xs:sequence></xs:complexType>
    </xs:element>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="SelecionarDptoRequest">
  <wsdl:part name="parameters" element="tns:SelecionarDpto"/>
</wsdl:message>
<wsdl:message name="SelecionarDptoResponse">
  <wsdl:part name="parameters" element="tns:SelecionarDptoResponse"/>
</wsdl:message>
<wsdl:message name="mainRequest">
  <wsdl:part name="parameters" element="tns:main"/>
</wsdl:message>
<wsdl:message name="SeleccionartemaRequest">
  <wsdl:part name="parameters" element="tns:Seleccionartema"/>
</wsdl:message>
<wsdl:message name="SeleccionartemaResponse">
  <wsdl:part name="parameters" element="tns:SeleccionartemaResponse"/>
</wsdl:message>
```

PortType o Interfaz de servicio:

```
<wsdl:portType name="SolicitudAprobPGAPortType">
    <wsdl:operation name="SelecionarDpto">
        <wsdl:input message="tns:SelecionarDptoRequest"/>
        <wsdl:output message="tns:SelecionarDptoResponse"/>
    </wsdl:operation>
    <wsdl:operation name="main">
        <wsdl:input message="tns:mainRequest"/>
    </wsdl:operation>
    <wsdl:operation name="Seleccionartema">
        <wsdl:input message="tns:SeleccionartemaRequest"/>
        <wsdl:output message="tns:SeleccionartemaResponse"/>
    </wsdl:operation>
</wsdl:portType>
```

Definición Concreta:

Binding:

```
<wsdl:binding name="SolicitudAprobPGASoap11Binding" type="tns:SolicitudAprobPGAPortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="SelecionarDpto">
        <soap:operation soapAction="urn:SelecionarDpto" style="document"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
    <wsdl:operation name="main">
        <soap:operation soapAction="urn:main" style="document"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
    </wsdl:operation>
    <wsdl:operation name="Seleccionartema">
        <soap:operation soapAction="urn:Seleccionartema" style="document"/>
        <wsdl:input>
            <soap:body use="literal"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="literal"/>
        </wsdl:output>
    </wsdl:operation>
</wsdl:binding>
```

Servicio:

```
  <wsdl:service name="SolicitudAprobPGA">
    <wsdl:port name="SolicitudAprobPGAHttpSoap11Endpoint" binding="tns:SolicitudAprobPGASoap11Binding">
      <soap:address location="http://localhost:8080/axis2/services/SolicitudAprobPGA.SolicitudAprobPGAHttpSoap11Endpoint/"/>
    </wsdl:port>
    <wsdl:port name="SolicitudAprobPGAHttpSoap12Endpoint" binding="tns:SolicitudAprobPGASoap12Binding">
      <soap:address location="http://localhost:8080/axis2/services/SolicitudAprobPGA.SolicitudAprobPGAHttpSoap12Endpoint/"/>
    </wsdl:port>
    <wsdl:port name="SolicitudAprobPGAHttpEndpoint" binding="tns:SolicitudAprobPGAHttpBinding">
      <http:address location="http://localhost:8080/axis2/services/SolicitudAprobPGA.SolicitudAprobPGAHttpEndpoint/"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

ANEXO 2

Definición Abstracta para el servicio Lista de Temas:

```
<wsdl:types>
  <xs:schema attributeFormDefault="qualified" elementFormDefault="qualified" targetNamespace="http://postgrado">
    <xs:element name="main">
      <xs:complexType>
        <xs:sequence>
          <xs:element maxOccurs="unbounded" minOccurs="0" name="args" nillable="true" type="xs:string"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
  </xs:schema>
</wsdl:types>
```

Mensajes:

```
<wsdl:message name="mainRequest">
  <wsdl:part name="parameters" element="ns:main"/>
</wsdl:message>
```

PortType o Interfaz de servicio:

```
<wsdl:portType name="ListaTemaPortType">
  <wsdl:operation name="main">
    <wsdl:input message="tns:mainRequest"/>
  </wsdl:operation>
</wsdl:portType>
```

Definición Concreta:

Binding:

```
<wsdl:binding name="ListaTemaSoap11Binding" type="ns:ListaTemaPortType">
  <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
  <wsdl:operation name="main">
    <soap:operation soapAction="urn:main" style="document"/>
    <wsdl:input>
      <soap:body use="literal"/>
    </wsdl:input>
  </wsdl:operation>
</wsdl:binding>
```

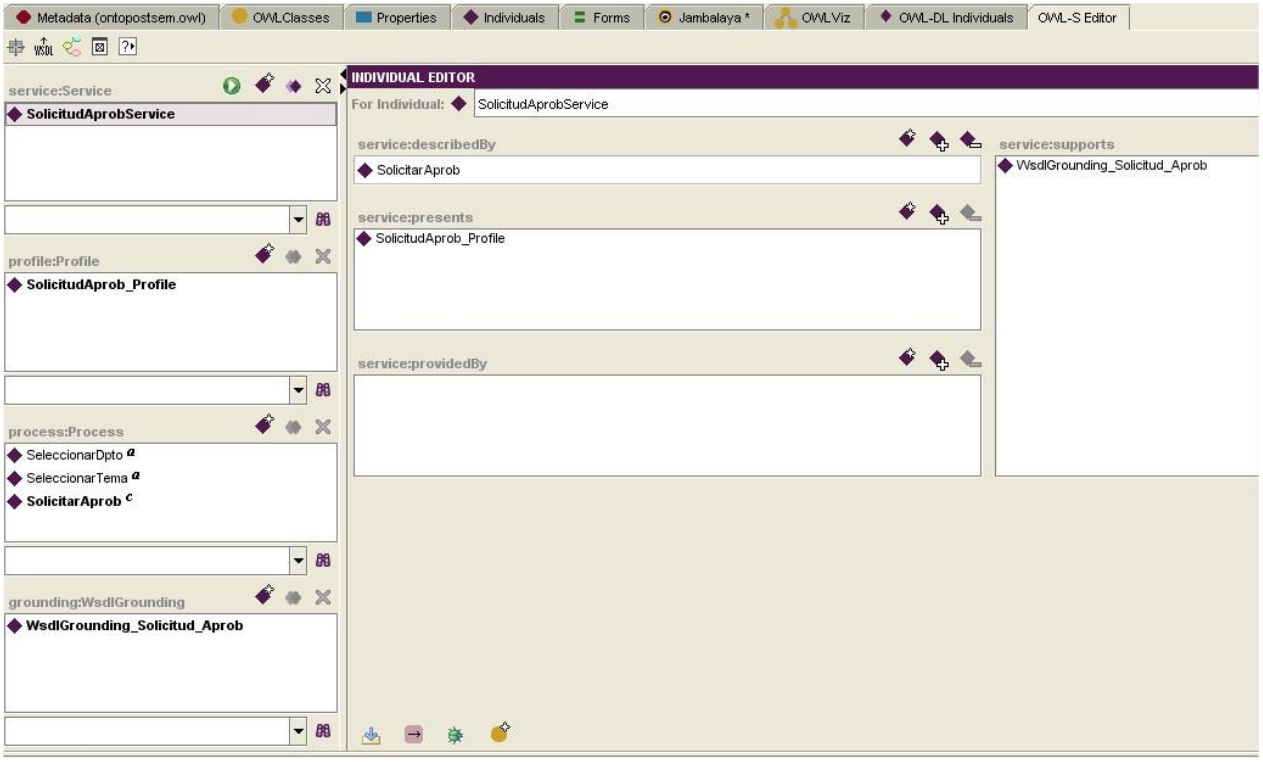
Servicio:

```
<wsdl:service name="ListaTema">
    <wsdl:port name="ListaTemaHttpSoap11Endpoint" binding="ns:ListaTemaSoap11Binding">
        <soap:address location="http://localhost:8080/axis2/services/ListaTema.ListaTemaHttpSoap11Endpoint/"/>
    </wsdl:port>
    <wsdl:port name="ListaTemaHttpSoap12Endpoint" binding="ns:ListaTemaSoap12Binding">
        <soap:address location="http://localhost:8080/axis2/services/ListaTema.ListaTemaHttpSoap12Endpoint/"/>
    </wsdl:port>
    <wsdl:port name="ListaTemaHttpEndpoint" binding="ns:ListaTemaHttpBinding">
        <http:address location="http://localhost:8080/axis2/services/ListaTema.ListaTemaHttpEndpoint/"/>
    </wsdl:port>
</wsdl:service>
```

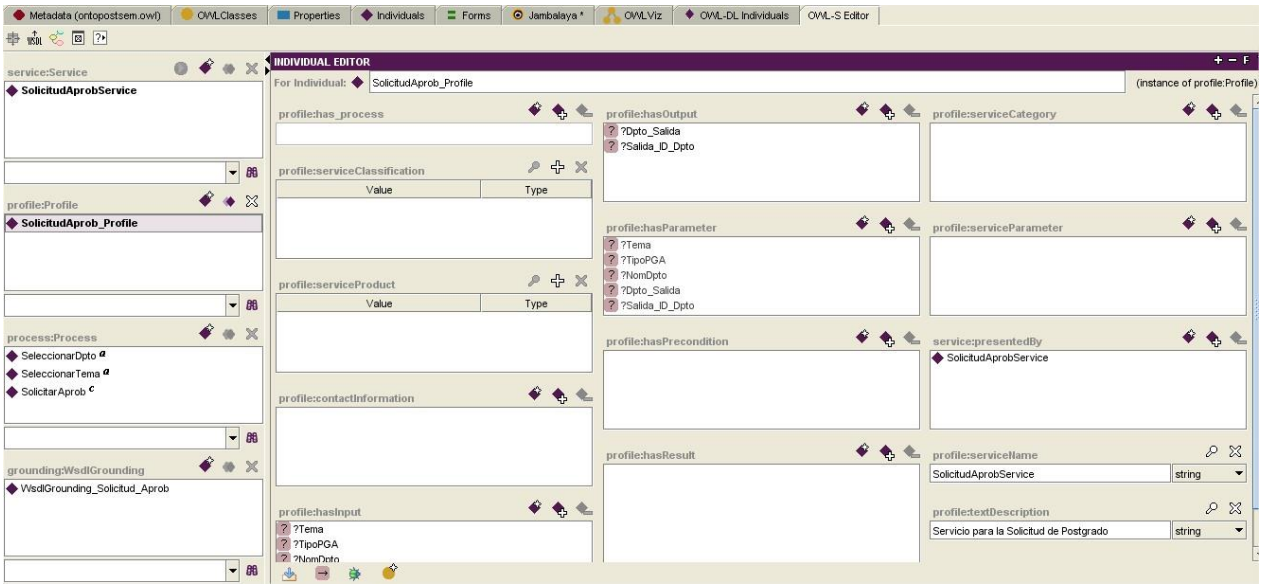
Los servicios lista de profesores, lista de departamentos, y lista de programas tienen la misma descripción que el explicado anteriormente lo que con otros atributos.

ANEXO 3

Interfaz de usuario para la creación de la instancia de una clase "Service" (SolicitudAprobService)



Interfaz de usuario para la creación de la instancia de una clase "*Profile*" y parámetros de entrada y salida (SolicitudAprob_Profile)



Interfaz de usuario para la creación de la instancia de una clase "*Grounding*" (WsdlGrounding)