TFC _ XML Web semántica

Memoria

Lenguajes de modelado de procesos para Servicios Web Semánticos. Caso de estudio: BPEL4WS y WSMO

Francisco Suárez Muiño
fsuarezmu@uoc.edu
I.T.I.S.

Sinuhé Arroyo Gomez
Sept. 2006 - Enero 2007

2. Resumen

En este documento se presenta un estudio acerca de las tecnologías implicadas en los servicios Web actuales, en consecuencia, se introducirá a la Web semántica y se extenderá a los servicios Web semánticos.

En este estudio se ha contextualizado las tecnologías actuales en los servicios Web y las nuevas tecnologías emergentes de los servicios Web semánticos. En consecuencia se han estudiado especialmente dos tecnologías: el Lenguaje de ejecución de procesos de negocio para servicios Web (BPEL4SW) perteneciente a los servicios Web y la Ontología de modelado para servicios Web (WMSO) que pertenece a los servicios Web semánticos. Posteriormente, se confecciona una comparación entre las dos tecnologías nombradas, fruto de esta comparativa se elaboraran unas conclusiones que fortalecen el estudio.

Se propone una ampliación del escenario para modelar. Se materializa el modelado tanto en WSMO como en BPEL4WS, aportando los códigos y diagramas correspondientes.

En la parte final del documento he adjuntado un glosario de términos donde se describen diferentes conceptos tratados durante los estudios.

La última sección de este análisis comprende una actualización de las principales direcciones de Internet que he utilizado para la obtención de los documentos base para realizar este documento.

3. Índice

4. Lenguajes de modelado de procesos para Servicios Web Semánticos.

4.1 Introducción

Para realizar los estudios, me he basado en las documentaciones oficiales interpretándolas y añadiendo los contenidos provenientes de otras fuentes que consideré necesarios hasta formar una composición que expusiese los puntos clave de las tecnologías, sobre todo en el caso de BPEL4WS y de WSMO.

4.1.1 Justificación del TFC y contexto en el que se desarrolla: punto de partida y aportación del TFC.

La Web, en su inicio, pretendía compartir documentos que se pudiesen referenciar entre sí, esto se conseguiría a partir de un sistema para nombrar de manera estándar la localización (URL) y donde el documento estuviese confeccionado con un lenguaje estándar que permitiese referenciar dichos documentos a través de marcas o hipertexto (HTML) y para poder realizar la transacción de información se crearía el protocolo HTTP.

Estos primeros pasos se realizan a principios de la década de los 90, a partir de este momento la Web evoluciona drásticamente, se permite incluir objetos multimedia en los propios documentos como imágenes, animaciones, sonidos, vídeos o elementos interactivos creados en diferentes lenguajes que se desarrollan de manera paralela.

Con el paso del tiempo los usuarios han crecido de manera espectacular, siendo únicamente en sus inicios el personal de investigación hasta llegar actualmente a superar el 16% de la población mundial. De la misma manera se ha elevado el número de contenidos, desarrollados en muchas ocasiones en diferentes idiomas y donde no solo se limitan a texto sino que se está incrementando la cantidad de elementos multimedia con mayor peso, posibilitado gracias a los adelantos tecnológicos, sobre todo de hardware.

Debido a este crecimiento y el poder de la red para elementos como el comercio o la educación, en los últimos años surgen los Servicios Web. Los Servicios Web permiten intercambiar información entre equipos de una red de manera independiente de las plataformas a través de una serie de protocolos estándar. Una de las principales ventajas es, a priori, la seguridad que se aporta al intercambiar los datos a través del puerto 80 del protocolo HTTP controlado por firewall. Una aplicación práctica de dichos Servicios sería la que aporta a dos empresas para que puedan interactuar,

encontrándose en zonas geográficas diferentes y que permite realizar operaciones de gran complejidad.

Para que la interactuación e interoperabilidad se realice de forma exitosa, deben de existir un serie de normas estándar y que a la vez estén abiertas a nuevas posibilidades, estos criterios están regulados por un consorcio llamado W3C dirigido por Tim Berners-Lee, que a su vez fue el creador original de los conceptos comentados anteriormente del inicio de la World Wide Web.

Existen diversos problemas derivados del formato original de la Web, por ejemplo la ineficacia de las búsquedas realizadas sobre el abundante contenido de la Web tradicional, además existe nuevo material multimedia que muchas veces no se está teniendo en cuenta en nuestra búsqueda. La clave de la ineficacia actual se encuentra en el código fuente, ya que se limita a mostrar las marcas que se interpretan a través de un visualizador. Por ejemplo si realizas una búsqueda de “hotel para estancia en París” un buscador emite entre sus resultados, por ejemplo, el de la oficina de turismo de París o de apartamentos para estancia en Berlín, además de los resultados que si son acerca de hotel en París. Por lo que se pueden observar que la búsqueda no es del todo eficaz.

Tim Berners-Lee, propone a través de la Web semántica, el enriquecimiento del lenguaje actual, donde el software tenga un concepto semántico de lo que está manejando ya que con la Web tradicional, las máquinas no comprenden que están mostrando. Al mismo tiempo se pretende la conexión entre diferentes recursos que se encuentren semánticamente relacionados. En resumen, se pretende una Web con mayor capacidad inteligente, que evolucione de la Web actual. Siguiendo el ejemplo anterior, que solo se muestren los resultados acordes con la búsqueda realizada.

Para llevar a cabo la Web semántica es necesario predefinir ciertos términos para que puedan ser analizados posteriormente por otras aplicaciones, son los llamados metadatos, en los que se dan detalles acerca de los objetos mostrados. Para obtener una definición se puede utilizar RDF (Infraestructura para Definición de Recursos) que es un modelo para la representación de los metadatos de manera simple.

Las ontologías, suponen un paso más, tienen como objetivo la definición básica y también sus correspondientes relaciones dentro de un área de conocimiento a través de una serie de lenguajes, son escalables, extensibles y abiertas. Permiten interactuar entre si semánticamente, en consecuencia, las búsquedas serán más precisas y relaciona las diferentes Servicios, bases de datos, dispositivos o recursos. Permiten conocer como se debe de etiquetar correctamente para el buen uso de los agentes automáticos.

Los Servicios Web también presentan los problemas propios de la Web tradicional y además problemas propios de las actividades que realizan, como puede ser el bajo rendimiento si hablamos en términos comparativos con la programación distribuida esto se traduce en una eficacia menor. Por otra parte no existe una estandarización total en

las interacciones entre los distintos Servicios, por lo que no se puede hacer una automatización “real” de Servicios.

En el caso de los Servicios Web semánticos se podrán buscar los propios Servicios y se realizará a través de tecnologías de reciente aparición como es la Web Service Modeling Ontology (WSMO) que se encarga de la corografía y orquestación entre distintos Servicios Web o el Business Process Execution Language For Web Services (BPEL4WS) que se encarga de la orquestación de los Servicios Web.

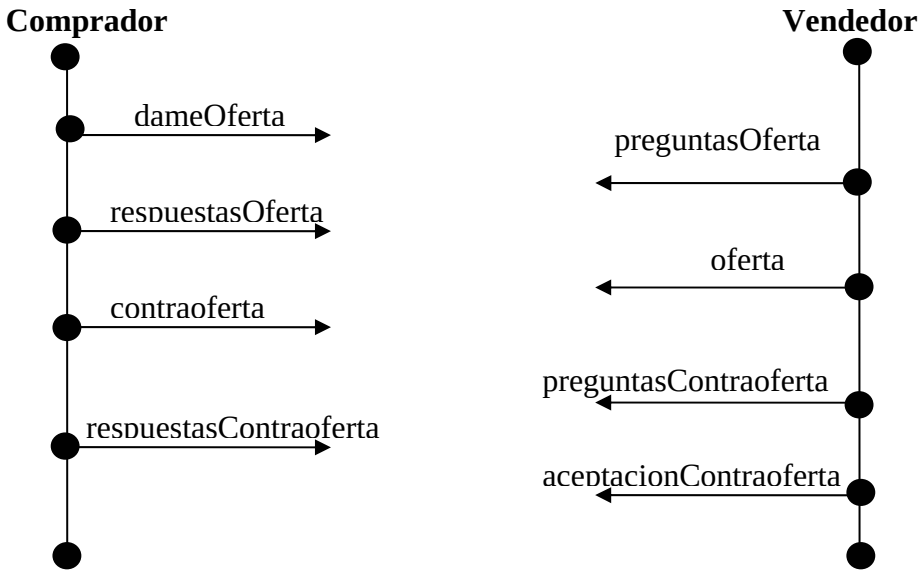
WSMO permite que los Servicios Web se realicen de manera automatizada, realizando a su vez su composición y ejecución, de manera que ofrece un apoyo conceptual y un lenguaje formal para describir semánticamente los aspectos relevantes de los Servicios Web, facilitando su automatización, combinación e invocación.

A través de BPEL4WS, podemos determinar en proceso de negocio, cuales son elementos con los que se relacionan desde que comienza el flujo de negocio hasta que presenta la salida, para orquestar dicho flujo, BPEL4WS decide cuando y que Servicio Web se ejecutará.

4.1.2 Objetivos del TFC.

Estudio de las tecnologías Web Service Modeling Ontology (WSMO) y Business Process Execution Language For Web Services (BPEL4WS) y análisis de WSMO y de BPEL4WS documentando los puntos clave de cada tecnología, a continuación se realizará una comparativa entre dichas tecnologías, prestando una especial atención a WSMO.

Dado el escenario propuesto:



Se realizará el modelado del escenario propuesto en WSMO y BPEL4WS estableciendo un caso concreto entre dos Servicios Web de ejemplo. Se ampliará el escenario para acometer el pago de la oferta aceptada, teniendo en cuenta las divisas.

4.1.3 Enfoque y metodología seguida.

Para la realización del trabajo se han tomado como referencia el cumplimiento de hitos como metodología, a través de los hitos se dará estructura y organización al proyecto presentado. Las fases de trabajo a cumplir son las siguientes:

4.1.3.1 Hito 1: Plan de trabajo

Se formaliza en la confección de este documento, en el que se concreta el alcance del Trabajo Fin de Carrera a realizar y la organización temporal que se va a llevar a cabo.

Se segmentan los hitos a completar para el desarrollo del proyecto.

Se recoge la primera documentación acerca de los elementos implicados y se contextualiza en base a mis conocimientos previos.

4.1.3.2 Hito 2: Análisis y comparativa

Dados los términos a manejar, en primer lugar se llevará a cabo un análisis de la documentación de manera minuciosa y una asimilación de conceptos acerca de la Web semántica, Servicios Web y en especial, acerca de los Servicios Web semánticos.

Dentro de estos conceptos se profundizará en la explicación de otras tecnologías como puedan ser Simple Object Access Protocol (SOAP), Universal Description, Discovery, and Integration (UDDI), Web Services Description Language (WSDL) o Web Services Interoperability (WSI)

En este hito se tendrán que estudiar extensamente las tecnologías WSMO y BPEL4WS para la aportación de los elementos claves a la hora de realizar la explicación del estudio y comparación de tecnologías.

Se realizará el documento que represente dichas introducciones y estudios para la entrega de la segunda prueba de evaluación continua donde se crearán los primeros términos del glosario.

4.1.3.3 Hito 3: Modelado

En este hito, dado el escenario propuesto, se llevará a cabo el modelado consecuente de seguir las pautas recogidas anteriormente. Dicho modelado se llevará a cabo tanto en BPEL4WS como en WSMO.

Además del modelado propuesto se añadirá una ampliación del modelado propuesto, en el que se tratará una transacción de pago después de la aceptación de la oferta teniendo en cuenta las divisas.

En caso de disponer del tiempo necesario, se procederá a realizar la implementación del modelado BPEL4WS presentado en este mismo hito.

Se añadirán términos al glosario.

Se redactará el informe con todas las conclusiones obtenidas durante este hito para su presentación en la tercera prueba de evaluación.

4.1.3.4 Hito 4: Memoria y presentación

Se procederá a realizar la memoria del proyecto teniendo en cuenta los puntos que se han definido anteriormente.

Se realizarán los cambios necesarios que certifiquen un modelado correcto si es que los hubiese.

Se revisan los términos de glosario, añadiendo alguno más si fuese preciso

Se realiza el documento de presentación que muestre el trabajo realizado a lo largo del semestre y los resultados obtenidos a través de una presentación en power point.

4.1.4 Planificación del proyecto.

4.1.4.1 Temporalización: calendario de tareas

A continuación se muestra el calendario con la planificación de las tareas de forma desglosada:

Diagrama de Gantt

	Nombre de tarea	Duración	Comienzo	Fin	Predecesoras
1	**– Hito 1: Plan de trabajo**	**9 días**	**vie 22/09/06**	**mié 04/10/06**	
2	Recogida de documentacion	3 días	vie 22/09/06	mar 26/09/06	3CC
3	Lectura y contextualizacion	8 días	vie 22/09/06	mar 03/10/06	
4	Redaccion del Plan de trabajo	8 días	vie 22/09/06	mar 03/10/06	2CC
5	Entrega de Plan de trabajo PEC1	1 día	mié 04/10/06	mié 04/10/06	4
6	**– Hito 2: Análisis y comparativa**	**27 días**	**jue 05/10/06**	**vie 10/11/06**	**5**
7	Análisis y asimilacion de la documentacio	12 días	jue 05/10/06	vie 20/10/06	5
8	Especificacion de otros conceptos	8 días	lun 23/10/06	mié 01/11/06	9CC;7
9	Intro y comparativa entre WSMO y BPEL	15 días	lun 23/10/06	vie 10/11/06	7
10	Redaccion del documento PEC2	15 días	lun 23/10/06	vie 10/11/06	8CC
11	**– Hito 3: Modelado**	**25 días**	**lun 13/11/06**	**vie 15/12/06**	**9**
12	Estudio de los requerimientos	5 días	lun 13/11/06	vie 17/11/06	10
13	Modelado del caso propuesto	10 días	lun 20/11/06	vie 01/12/06	12
14	Implementacion (opcional)	10 días	lun 04/12/06	vie 15/12/06	13
15	Redaccion del documento PEC3	20 días	lun 20/11/06	vie 15/12/06	12CC;13CC;14FF
16	**– Hito 4 Memoria**	**19 días**	**lun 18/12/06**	**jue 11/01/07**	
17	Revision del modelado	8 días	lun 18/12/06	mié 27/12/06	15
18	Testing	6 días	jue 28/12/06	jue 04/01/07	17
19	Notas para memoria	8 días	lun 18/12/06	mié 27/12/06	15
20	Confección de la presentación	8 días	jue 28/12/06	lun 08/01/07	19
21	Preparacion de la presentacion	3 días	mar 09/01/07	jue 11/01/07	20
22	Redaccion de la memoria	19 días	lun 18/12/06	jue 11/01/07	15

Se han definido las siguientes fechas para el desarrollo de la planificación del proyecto, coincidiendo con los hitos: plan de trabajo, análisis y comparativa, modelado y memoria.

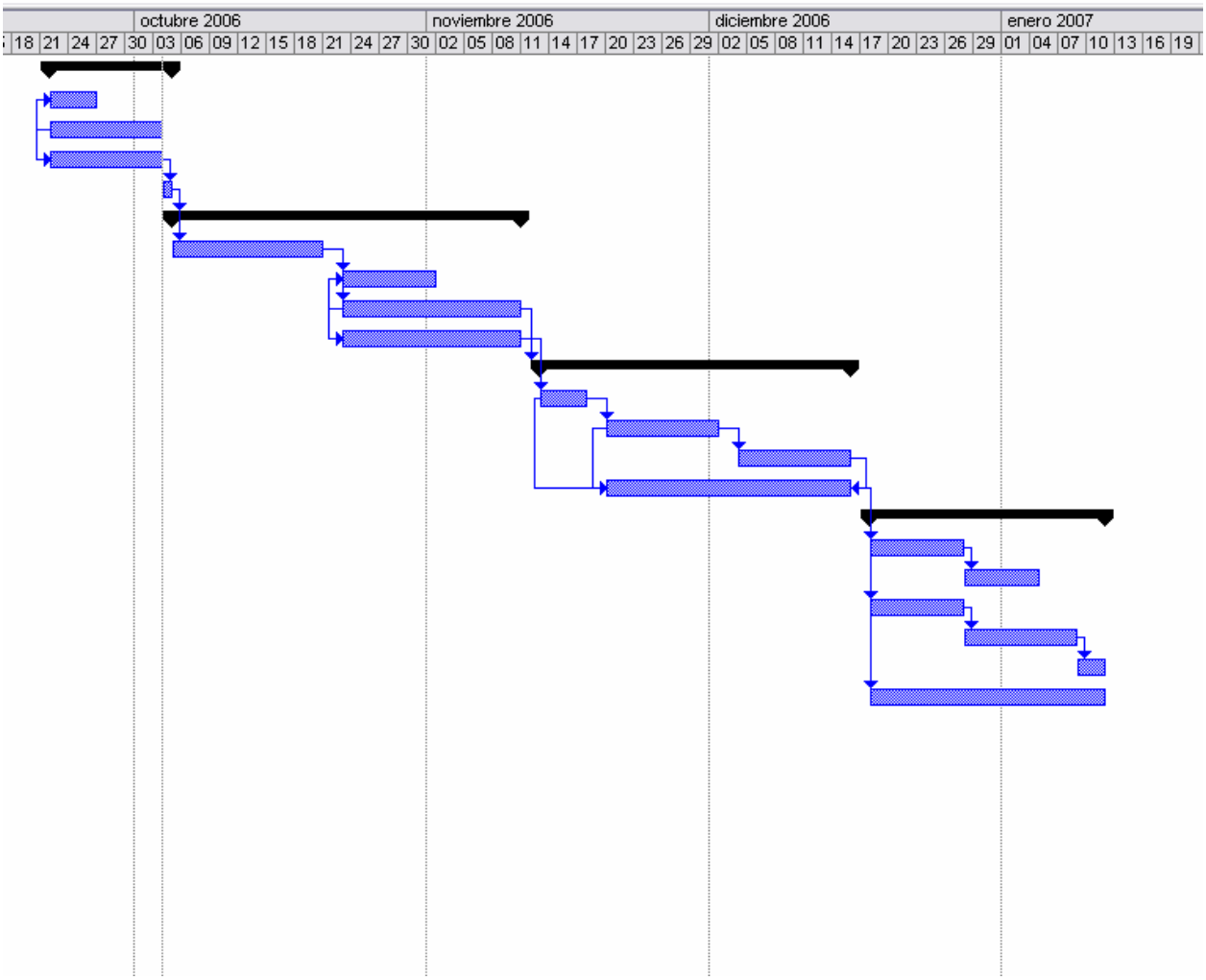
- Plan de trabajo: Del 22 de Septiembre al 4 de Octubre con las tareas descritas en el Hito 1 y que termina con la confección del documento Plan de trabajo.

- Análisis y comparativa: Desde 5 de Octubre hasta el 10 de Noviembre, se realizarán los puntos detallados en el Hito 2 y culminará con la entrega del documento acerca del análisis y de la comparativa requerida.

- Modelado: Entre el día 11 de Noviembre y hasta el 15 de Diciembre, donde se realizará el modelado de cada uno de los apartados detallados en el Hito 3.

- Memoria: A partir del 16 de Diciembre y antes del 11 de Enero se procederá a la entrega del documento de la memoria final y de la presentación del proyecto, después de haber realizado las modificaciones necesarias para su correcta aprobación, como se indica en el Hito 4 de este documento.

4.1.4.2 Diagrama de Gantt



4.1.5 Productos obtenidos.

En cuanto a los resultados obtenidos, se puede interpretar en primer lugar todas las consideraciones consecuentes de la comparativa entre las tecnologías ofreciendo nuevos puntos de vista para las tecnologías WSMO y BPEL4WS, tal y como se muestra en el punto 4.2.4.

Además de las especificaciones técnicas, fruto de su comparativa, se confeccionan una serie de conclusiones y líneas de futuro para estas tecnologías que pertenecen a los puntos 4.3 y 4.4 respectivamente.

Otra parte a destacar surge a través del modelado del escenario propuesto. Con el modelado de BPEL4WS (punto 4.2.5.2) se obtienen diagramas y la codificación correspondiente y a través del modelado de WSMO (punto 4.2.5.3) también se puede obtener un acercamiento a la tecnología en el caso presentado y posteriormente la codificación correspondiente.

4.1.6 Breve descripción de los otros capítulos de la memoria

Se puede comentar la contextualización que se realiza de las tecnologías estudiadas. En un principio se comienza con el estudio de los servicios Web, en el punto 4.2.1, destacando las tecnologías más importantes y las herramientas más comunes. Se realiza lo mismo con la Web semántica (punto 4.2.2) y los servicios Web semánticos (punto 4.2.3).

En el estudio de las tecnologías de los servicios Web se hace referencia además de BPEL4WS en el punto 4.2.1.1.3 que se realiza de manera ampliada, a SOAP en el punto 4.2.1.1.1, a WSDL en el punto 4.2.1.1.2 y a UDDI en el punto 4.2.1.1.3.

En el estudio de las los servicios Web semánticos, he analizado la tecnología WSMO, en el punto 4.2.3.1.1, profundizando en los aspectos principales de dicha tecnología como son las ontologías, los mediadores, las metas y los servicios Web.

4.2 Proyecto

4.2.1 Servicios Web

4.2.1.1 Tecnologías

Las tecnologías para servicios Web surgen como la evolución de los primeros aplicaciones distribuidas, que posteriormente se desarrollaron tecnologías poco flexibles como fueron DCOM, CORBA, RMI... Las tecnologías para servicios Web se basan en una aplicación que es accesible a través de protocolos Web como puede ser http para ofrecer una información solicitada a través de lenguajes estándar como XML.

A continuación se detallan las tecnologías que intervienen en el desarrollo de los servicios Web, como son SOAP, UDDI y WSDL. Se explicará también la tecnología BPEL4WS que proporciona el modelado de los servicios Web.

4.2.1.1.1 SOAP

Es el acrónimo de Simple Object Access Protocol y es un protocolo de servicios Web que define la manera de comunicar dos objetos por medio del intercambio de mensajes basados en XML facilitando en comparación con otros protocolos la lectura de los mensajes y contando con las ventajas del uso de XML.

SOAP está creado por varias empresas importantes como puede ser Microsoft o IBM entre otras y está apoyado por la W3C.

Un mensaje SOAP en su creación se puede comparar con un sobre y que consta de una cabecera y de un cuerpo de mensaje. En la cabecera se indica como se tiene que procesar el mensaje además de otras informaciones adicionales. Luego el cuerpo del mensaje se mantiene intacto hasta que llega a su destinatario, pudiendo pasar por diversos intermediarios.

4.2.1.1.2 WSDL

Sus siglas significan Web Services Description Language y se usa como estándar para describir la manera de comunicación de los servicios Web a través de un lenguaje en XML.

Este lenguaje es necesario para lograr la comunicación entre dos objetos que a priori, se desconocen. Para lograr una comunicación, primero se presentan los tipos de datos, formatos y protocolos por parte de un objeto, de esa manera lo podemos clasificar si es necesario para que se pueda dar a conocer en caso de que sea de forma automática.

Su estructura puede resumirse como una lista de definiciones donde además de la propia definición del WSDL, nos encontramos con otros bloques:

- Bloque de documentación: donde tenemos la posibilidad de añadir y extender referencias de documentación.

- Bloque de Inclusión: donde podemos importar otros documentos WSDL ya sea por importación o por inclusión.

- Bloque de definición de tipos: donde podemos especificar de forma abierta las estructuras de datos necesarias.

- Bloque de interfaz: donde se especifican las operaciones que soporta el servicio Web, además se pueden especificar la gestión de las excepciones y las características y propiedades de las operaciones.

- Bloque de enlaces: especifica de manera concreta la manera de enviar mensajes para la comunicación.

- Bloque de servicios: donde se describen los puntos de entrada unidos a la dirección física de los servicios Web.

WSDL está apoyado por la W3C y está aceptado de manera mayoritaria.

En el entorno de los servicios Web, tiene como limitación que no se describe el formato y el papel o rol del mensaje en la interacción de los servicios Web.

Tiene como ventajas la flexibilidad, ya que está abierta a nuevos elementos, pero también una restriccion desde el punto de vista de los servicios Web, ya que no hay una semántica para las secuencias de mensajes, la correlación y el contenido.

Otro aspecto a tener en cuenta, es que WSDL, por si solo, no nos dice que proveedor nos da un servicio, ni que interacción hace, ni cual es la secuencia de operaciones.

4.2.1.1.3 UDDI

Son las siglas de Universal Description, Discovery, and Integration y viene a ser un servicio de directorios para servicios Web basado en XML.

Está considerado como un estándar en los servicios Web para que a través de mensajes SOAP puedan obtenerse documentos WSDL donde se especifican los requisitos del protocolo y el formato del mensaje solicitado.

La filosofía de se basa tener un sistema de clasificación de servicios según unas características como son la descripción de los propios servicios, incumbiendo también a

las compañías, tiene que ser flexible y abierto a un futuro con nuevas posibilidades, teniendo que poderse acceder siempre a lo ya guardado. Por último, tiene que poseer una integración de servicios para poder invocarlos.

En su arquitectura se pueden encontrar tres niveles:

- Páginas blancas, con direcciones, contactos e identificadores conocidos.
- Páginas amarillas, con categorizaciones industriales.
- Páginas verdes, donde a partir de servicios ofrecidos por empresas, se obtienen informaciones técnicas.

UDDI que está apoyada por OASIS y en la versión 2.0 presenta limitaciones en la funcionalidad de las búsquedas, ya que solo busca una palabra clave.

4.2.1.1.4 BPEL4WS

El lenguaje de ejecución de procesos de negocios para servicios Web proporciona un lenguaje para la especificación formal de procesos y protocolos de la interacción de negocio, que extiende al modelo de interacción de los servicios Web definido por WSDL.

Es un lenguaje independiente, en cuanto a la plataforma de ejecución y teóricamente portable.

BPEL4WS describe una capacidad conocida comúnmente como orquestación que es un componente dominante de las arquitecturas orientadas a servicio (SOA), y que permite a los procesos del negocio estar basados en servicios reutilizables, de manera que ahorra coste y resulta más ágil.

Este lenguaje está basado en el estándar XML y especifica de manera formal, cuales son los procesos de negocio y cuales son los protocolos de interacción de manera que se especifica como se conectan los servicios Web entre si para realizar una determinada tarea.

En la orquestación, las estructuras de control son secuenciales, concurrentes e indeterministas. Están especificadas en XML.

En cuanto a los datos y su acceso, estos están descritos en XML Schema 1.0 y existen los tipos de datos de control, correlación y los específicos según la propia aplicación. El acceso a los mismos se realiza a través de Xpath 1.0 y permite la integración de extensiones en XML.

Los pilares en los que se fundamente BPEL4WS, además de XML, son SOAP y WSDL, debido a la fiabilidad de estos elementos, se puede desarrollar complicados escenarios de servicios Web que presenten procesos de negocio.

BPEL4WS será la base principal para la orquestación de servicios Web a través de las especificaciones necesarias para describir de manera formal los procesos de negocio, su interoperabilidad e interacción.

Existen process que son procesos de negocio que invocan a servicios Web para realizar tareas funcionales, estos procesos de negocio pueden ser descritos de dos modos:

- Procesos de negocio ejecutables, modelan el comportamiento real de un usuario en una interacción de negocio, están especificados para ser ejecutados.

  Un proceso ejecutable se especifica completamente y se puede ejecutar por un motor apropiado, ya que en ellos se encuentran todos los pasos de ejecución. Son procesos privados.

- Procesos de negocio abstractos, son procesos parcialmente especificados que no son requeridos para ser ejecutados pero que tienen un papel descriptivo. Pueden ocultar detalles requeridos por procesos ejecutables.

  Los procesos abstractos describen que puede hacer un proceso, sus entradas y salidas pero no describe como consigue realizarlo, son útiles para describir procesos de negocios a otros que los deseen utilizar. Se puede, en cierto modo, pensar en como si fueran plantillas que demuestren la funcionalidad de proceso que se requiere que pero que no se especifica completamente. Son procesos públicos.

El lenguaje BPEL4WS define un modelo y una gramática para describir el comportamiento de un proceso de negocio basado en interacciones entre el proceso y otros elementos o socios. La interacción con cada compañero ocurre a través de interfaces de Servicio Web y la estructura de la relación en el nivel de interfaz está encapsulada (un partnerLink).

Hay cuatro partes importantes a tener en cuenta en un proceso de negocio:

- partnerLinks: definen los elementos que interactúan con el proceso de negocio, cada partnerLink es una instancia concreta de interacción en la que se describen los mensajes intercambiados y el protocolo de transporte, está caracterizado por un partnerLinkType, este identifica la funcionalidad que debe de proporcionar el proceso de negocio y su socio para que la relación tenga éxito. Son interacciones abstractas a entre servicios desempeñando diferentes roles.

- variables: definen las variables de datos usadas en los procesos, sirven para mantener el estado en los intercambios de mensajes.

- faultHandlers: donde se encuentran las actividades que se deben de realizar en caso de fallo en los resultados de las invocaciones. Todos los fallos son identificadas por un nombre.

- process: contiene la descripción del comportamiento normal para manejar una petición.

En la orquestación de las interacciones existen una serie de tipos de control:

- Secuencial a través de sequence, switch y while

- Concurrente y sincronizado mediante flor.

- No determinista que está basado en eventos a través de Pick.

Un proceso está formado por una o varias actividades unidas por links, el camino que toma las actividades viene determinado por varios aspectos, incluyendo las variables de evaluación de expresiones.

Los puntos de partida se llaman start activities y crean los atributos de instancia, entonces, cuando se acciona una start activity se crea una nueva instancia de proceso de negocio. A partir de ahora la instancia estará identificada por los correlation sets, que identifican un proceso pero que pueden cambiar después de un tiempo. Por ejemplo, un número de pedido, puede tener una correlación por ser el comienzo, pero luego al generar la factura la identificación por correlación puede variar.

En BPEL4WS también están contempladas las excepciones, a través de un sistema basado en try-catch-throw que se ejecuta cuando se activa algún manejador de fallos.

BPEL4WS tiene un ciclo de vida, en el cual puede haber largas interacciones con su inicio, su ciclo de vida y un final.

4.2.1.2 Herramientas

Dadas la magnitud de los servicios Web existe una mayoría de empresas que proporcionan herramientas para desarrollo de servicios Web, por ejemplo IBM, Oracle, Microsoft, Sun... Cada una presenta sus herramientas propias de desarrollo.

Se puede destacar en el caso de BPEL4WS la apuesta de Active por las herramientas de uso libre como puede ser el Active BPEL designer que puede funcionar sobre el Active BPEL engine. Grandes empresas como Oracle ofrecen en la misma línea el BPEL process manager.

4.2.2 Web semántica

4.2.2.1 Tecnología

A partir de la implantación de la Web semántica tanto las personas como los ordenadores conocerán de qué están tratando en cada momento, proporcionando un sentido inteligente a la World Wide Web, con una automatización de tareas, razonamiento y gestión del conocimiento.

Para que esta nueva visión se lleve a cabo hay tres tecnologías imprescindibles, son XML, RDF y las ontologías.

XML ya se utiliza en la Web actual para el intercambio de datos, y seguirá siendo un punto básico en la Web semántica debido a su universalidad. XML funciona a través de URIs por lo que se puede asegurar que se nombran de forma única cada recurso de la Web semántica.

A través de RDF y también RDFS, podremos añadir un significado a los elementos, incorporando metadatos sin interferir en la estructura del propio elemento. De esta manera se especifica la semántica para los datos que se basan en XML

Por último necesitamos a las ontologías, que describen a través de metainformaciones la semántica de un elemento, teniendo que estar expresadas en un formato legible por los ordenadores. Proporciona un lenguaje de descripción más avanzado y complejo que los anteriores.

4.2.2.2 Herramientas

En cuanto a las herramientas propias de los elementos de la Web semántica cabe destacar los que tienen como propósito la gestión y edición de ontologías.

Se puede destacar el Protégé que es de código abierto y permite definir ontologías en RDFS y posteriormente su exportación. Existen extensiones para otras tecnologías complementarias.

4.2.3 Servicios Web semánticos

4.2.3.1 Tecnologías

A través de la tecnología WSMO se intenta dotar de sentido semántico a los servicios Web, a través de su automatización y composición, a continuación se explicará extensamente.

Para la automatización de tareas, se usa otra tecnología basada en el lenguaje WSML que se llama WSMX que es un entorno de ejecución de servicios Web semánticos.

Otra tecnología relacionada es el Semantic Web Services Lenguage SWSL que se usa para la descripción de las ontologías de los servicios Web semánticos.

4.2.3.1.1 WSMO

La ontología de modelado para servicios Web, proporciona las especificaciones ontológicas para los elementos base de los servicios Web semánticos.

WSMO se basa en los siguientes principios básicos del diseño:

- Conformidad del Web: WSMO hereda el concepto de URI para la identificación única de recursos como el principio esencial. Por otra parte, WSMO adopta el concepto de Namespaces para denotar espacios constantes de la información, se apoya en XML y otras recomendaciones de la tecnología Web de W3C, así como la descentralización de recursos.
- Está basado en ontologías: Los datos modelados por medio de WSMO, se forman a través de ontologías, aportando un significado a todas las descripciones siendo una parte central de la Web semántica.

- Desemparejamiento estricto: Los recursos de WSMO están definidos y especificados de manera aislada, es decir, cada recurso es independiente tanto en su uso como en la interacción con otros recursos.

- Centrado en mediación: WSMO reconoce la importancia de la mediación para el correcto desarrollo de los servicios Web, haciendo que la mediación sea un componente de primera clase.

- Separación de roles ontológicamente: Se trata de la diferenciación entre los deseos de los usuarios de un servicio Web y la disponibilidad que presentan, pudiendo en algunos casos no ser los mismos.

- Descripción vs. implementación: WSMO diferencia entre las descripciones formales requeridas y las tecnologías que permiten su implementación, teniendo

que proporcionar un modelo ontológico adecuado a una descripción que pueda ser implementado por las tecnologías existentes o por otras emergentes.

- Semántica de ejecución: La referencia de WSMX y otros sistemas permitidos por WSMO permiten verificar la semántica y por lo tanto la especificación WSMO en su ejecución.

- Servicio vs. Web servicios: WSMO, pone los medios para describir los servicios Web para acceder a los propios servicios, sin sustituir al propio servicio.

En los aspectos relacionados con los servicios Web semánticos, WSMO es un meta-modelo especificado a través de la Meta-Object Facilty (MOF) y define 4 capas en la arquitectura de los meta-datos:

- Capa de información: abarca los datos e información que se describen. En esta capa se pueden englobar los datos descritos por las ontologías intercambiados por los servicios Web.

- Capa de modelo: abarca los meta-datos descritos por los datos de la capa de información. A esta capa pertenecen las ontologías, los servicios Web, las metas y las especificaciones de los mediadores.

- Capa meta-modelo: abarca las descripciones de la estructura y semántica de los meta-datos. WSMO en si mismo, se encuentra dentro de esta capa.

- Capa meta-meta-modelo, abarca las descripciones de la estructura y semántica de los meta-meta-datos. Se puede englobar aquí la lengua que define WSMO

Cada elemento WSMO está identificado por:

• Referencias URI: es una de las bases de WSMO, se identifican las entidades a través de los identificadores del Web, por lo que todo está denotado por un URI, excepto si se clasifica identificador anónimo.

• Identificadores anónimos: se usan para denotar elementos que existen pero que no necesitan tener un identificador especifico, puede ser una identificación numerada o no numerada, tienen como alcance el de una expresión lógica.

Los literales son utilizados para identificar valores a través de una identificación léxica y los tipos de datos.

Existen unos elementos a nivel superior de WSMO son los siguientes:

- Las ontologías: proporcionan la terminología utilizada por otros elementos de WSMO para definirse.

- Los servicios Web: proporcionan el acceso a los servicios dentro de un dominio, abarcando capacidades, interfaces y el funcionamiento del servicio Web usando la terminología definida con las ontologías.

- Las metas: son los propios deseos del usuario, lo que quiere que se cumpla a través de un servicio Web

- Los mediadores: son los que se encargan de resolver aquellos problemas por incompatibilidades presentadas en el nivel de datos, por ejemplo: resolviendo uniones mal realizadas por las terminologías o combinando servicios Web y las metas.

4.2.3.1.1.a Las ontologías

Son la clave para ligar la semántica conceptual que existe en el mundo real y la que está convenida y definida por comunidades de usuarios. Podemos decir que define una terminología común proporcionando conceptos y relaciones entre conceptos. Se proporcionan además unos axiomas que son expresiones en una lengua lógica.

Las ontologías describen toda la información necesaria para automatizar los descubrimientos, las composiciones, las ejecuciones...

Se pueden importar ontologías para construir algunos dominios de manera modular, de modo que se reduce la complejidad para la construcción de ciertos dominios.

Podemos tener que alinear las ontologías cuando son importadas, para ello, se usan los mediadores, conformes a los principios básicos del diseño.
Los conceptos son los elementos básicos de la terminología convenida para un dominio del problema, estos tienen cualidades, nombres y tipos.

Las relaciones son para modelar las interdependencias de varios conceptos.

Las funciones son tipos especiales de relaciones, en las que hay una semántica definida que permite comprobar si los valores de entrada se producen para unos ciertos parámetros.

Las instancias, dentro de las ontologías, están definidas explícitamente.

4.2.3.1.1.b Los servicios Web

Un servicio Web es una entidad capaz de invocar uno o varios objetivos propuesto por los usuarios, un servicio es el valor proporcionado por esta invocación.

Está formado por los siguientes elementos:

• Capacidad: que describe al servicio Web mediante su funcionalidad, una capacidad de un servicio Web está definida por la especificación de los siguientes elementos: propiedades no funcionales, ontologías importadas, mediadores usados, variables compartidas, precondiciones, poscondiciones y efectos.

• Interfaz: que describe como se puede alcanzar la funcionalidad del servicio Web, lo realiza a través de propiedades: la coreografía y la orquestación.

⊙ La coreografía que define cómo se comunica con el servicio Web para alcanzar una funcionalidad.

En WSMO se realiza desde el punto de vista del cliente, pudiendo definirse coreografías múltiples. El modelo conceptual seguido es el de las Abstract State Machines, que proporciona un grado alto de flexibilidad y que presentan los States y los Guarded Transitions.

Los estados (States) pueden presentar una propiedad de modo, que especifica una característica nueva no funcional, pudiendo tener los siguientes valores:

- Static: si la extensión de concepto, la relación o función no pueden ser cambiada, es el valor por defecto.
- Controlled: si la extensión de concepto, relación o función solo puede ser cambiada por el servicio Web.
- In: especifica que la extensión de concepto, relación o función puede ser cambiada por el entorno y se debe de proveer de un acceso de escritura para el entorno.
- Shared: especifica que la extensión de concepto, relación o función pueda ser cambiada por el entorno y por el servicio Web, tendríamos que proveer de acceso con permisos de lectura y escritura en el entorno.
- Out: que especifica que la extensión de concepto, relación o función puede ser cambiada por el servicio Web y se debe de proporcionar el acceso de lectura para el entorno.

Las Guarded transitions expresan cambios de estados mediante reglas condicionales de tipo if...then.

⊙ La orquestación define como se alcanzan las funcionalidades finales a través de varios componentes esenciales de servicio, es decir como se emplean otros servicios para alcanzar la capacidad o funcionalidad requerida, siendo el modelo de interacciones que un agente del servicio Web tiene que seguir

para alcanzar un objetivo.

En WSMO, la orquestación tiene al igual que la coreografía States y Guarded transitions, pero estas últimas, a diferencia de en las coreografías, en el then se apoya en mediadores para unir la orquestación para otras metas o servicios Web. Los mediadores usados pueden ser:

- wwMediator: si se conoce el servicio Web requerido, uniendo la orquestación a la coreografía del servicio Web requerido.
- wgMediator: si el servicio Web es desconocido, ya que se vincula a la meta requerida por la orquestación en el estado dado.

4.2.3.1.1.c Metas

Las metas son la representación de un objetivo que se obtiene por la ejecución de un servicio Web. Estas metas pueden ser las descripciones de un servicio Web que busca el usuario.

En WSMO las metas están descritas por propiedades no funcionales, ontologías importadas, mediadores usados, capacidades requeridas e interfaces requeridas.

Una meta puede importar ontologías existentes a través del propio conocimiento ontológico o bien a través de mediadores.

4.2.3.1.1.d Mediadores

Gracias a la existencia de mediadores se permite unir dos recursos heterogéneos, ya que resuelven las incompatibilidades en diferentes niveles:

- Nivel de datos: mediando entre diferentes terminologías, resolviendo el problema de la integración ontológica.
- Nivel de proceso: mediando entre patrones heterogéneos de comunicación que se dan en la comunicación entre diferentes servicios Web.

Existen cuatro tipos de mediadores:

- GGMediators: son aquellos que unen dos metas.
- OOMeditators: son aquellos que importan ontologías y resuelven una posible representación entre ontologías desemparejadas.

- WGMediators: son los que unen servicios Web a las metas, de manera que el servicio Web realiza totalmente o parcialmente el objetivo requerido. Esta unión puede ser a través de orquestación o coreografía.

- WWMediators: unen a dos servicios Web

4.2.3.1.2 Herramientas

Las herramientas para los servicios Web semánticos están en un proceso de maduración y de momento hay pocas opciones disponibles.

Como más importante destaca el WSMO Studio que facilita la edición de ontologías en el modelado de servicios Web semánticos. Otra herramienta similar es DOME, DERI ontology management enviromenment.

A través de WSMO4J podemos desarrollar aplicaciones siendo un API y una referencia para el desarrollo de aplicaciones de servicios Web semánticos, siguiendo los estándares marcados por WSMO y WSML. También realiza funciones de validación de WSML.

WSML Reasoner GUI, permite la edición como herramienta razonadonadora en la implementación de WSML.

4.2.4 Comparativa entre BPEL4WS y WSMO

4.2.4.1 BPEL4WS vs. WMSO

Después del estudio de las características de las principales tecnologías de servicios Web, puedo llegar a la conclusión que BPEL4WS y WMSO son comparables parcialmente, debido a que parten de diferentes orígenes para ofrecer soluciones a un problema.

El concepto angular que hace que los orígenes comentados sean distantes viene dado por la semántica. WMSO se centra en la Web semántica para dar solución al mismo problema que BPEL4WS soluciona desde un punto de vista que carece de semántica en su origen.

Entremos en conceptos más específicos de las tecnologías:

WSMO, al tener un origen semántico, cuenta con las siguientes ventajas respecto a BPEL4WS:

- Localiza varios servicios convenientes de manera semántica para una tarea dada, seleccionando el más conveniente entre los disponibles. Es la única tecnología que se ocupa de la coreografía desde un punto de vista semántico, por lo que las metas describen el comportamiento de los servicios desde el enfoque de los usuarios

- Compone los servicios para alcanzar la meta deseada orientando las búsquedas de forma inteligente, incluso a través de mediadores, resolviendo desencuentros de datos, protocolos o procesos.

- Invoca los servicios para su ejecución controlando el proceso de ejecución teniendo la ventaja de sustitución de servicios por otros equivalentes.

- En la publicación, dispone de la descripción de la funcionalidad del servicio.

BPEL4WS debido a la carencia de semántica, tienen otra manera de enfocar el problema:

- Describe como se componen varios servicios Web a partir de una estructura de servicio Web en términos pasos en los procesos individuales, enlaces de flujos de datos y control de enlaces de flujos.

- Tiene interacción entre el proveedor y el demandante, también entre las operaciones internas y los tipos de puertos WSDL pero está limitado por la nula descripción de entradas, salidas, precondiciones y efectos de los servicios Web y solo se pueden hacer composiciones de forma “manual”.

- Presenta problemas para descubrir resultados cuando se realizan las peticiones en diferentes términos entre la parte demandante y la oferta disponible.

- Pueden darse dificultades para interpretar los resultados devueltos por los servicios Web, debido a la falta de semántica. No define mecanismos para alinear intercambios heterogéneos de mensajes, ni describe métodos y técnicas para confirmar la compatibilidad entre mensajes.

WSMO puede presentar la desventaja frente a BPEL4WS acerca de la manera de regresar la información después de hacer una composición de servicios, debido a los tiempos de respuesta pueden ser desconocidos en algunos casos. BPEL4WS contempla desde un origen capacidad para gestionar múltiples servicios síncronos y asíncronos, dentro de los flujos de proceso colaboradores y transaccionales.

Otra característica en contra de WSMO, se presenta porque no separa los diferentes modelos implicados en su coreografía, al estar basado en ASM sin indicar claramente si se apoyan en aspectos de comportamiento o estructurales, entonces, se puede desarrollar una especificación difícil de entender o interpretar.

WSMO no es una tecnología madura, a pesar de estar apoyada por la W3C y tener distintas iniciativas en funcionamiento, hay puntos que se tienen que mejorar y BPEL4WS si es goza de mayor experiencia y el apoyo de grandes empresas como IBM, Microsoft, BEA, SAP u Oracle, pero en origen se basa en varias tecnologías con fuente en un XML rudimentario sin contenido semántico.

Por la misma razón, al ser semántico, WSMO describe las especificaciones de las metas, pero las técnicas y los mecanismos para compatibilizar los mensajes en los intercambios no están muy desarrollados ni definidos y pueden llegar a ser un problema.

Si a partir de ahora nos centramos en el funcionamiento, WSMO se concentra en el problema de la interoperabilidad, intentando solucionar el problema de la integración de las aplicaciones para servicios Web, a pesar de incluir de forma importante aspectos relacionados con el descubrimiento, la composición y la invocación. Por ello hay una especial importancia de los mediadores en WSMO, tanto para las ontologías como para las metas, servicios, y metas con servicios, solucionando de ese modo la interoperabilidad, BPEL4WS se diseña para manejar flujos de proceso a través de los puntos finales de los servicios Web, centrándose especialmente en la orquestación.

En la fase de descubrimiento, WSMO distingue los puntos de vista del proveedor y del demandante e introduce dos conceptos diferentes, las metas y las capacidades. Las metas son los objetivos de los clientes cuando realizan una consulta a un servicio Web y las capacidades es la funcionalidad que el servicio proporciona. Se pueden asociar varias metas usando mediadores y solo a una capacidad. En comparación, este supuesto puede estar solucionado en el caso de usar BPEL4WS a través de una

tecnología como WSDL-S, ya que por especificaciones en origen, BPEL4WS carece de los servicios de descubrimiento.

En la interoperación, BPEL4WS tendría que desarrollar lo descrito por WSDL-S, para proceder a realizar el modelo abstracto, mientras en WSMO se definen los servicios y las capacidades. En ambos casos se tienen en cuenta las entradas y salidas, así como los efectos.

En cuanto a la composición en WSMO se describe la coreografía y la orquestación basada en la tecnología de Abstract State Machines, WSMO usa múltiples interfaces de coreografía y orquestación, BPEL4WS usa el mecanismo basado en el modelo de los procesos abstractos

En la invocación, WSMO introduce mecanismos que tienen su base en WSDL mientras que BPEL4WS utiliza directamente WSDL

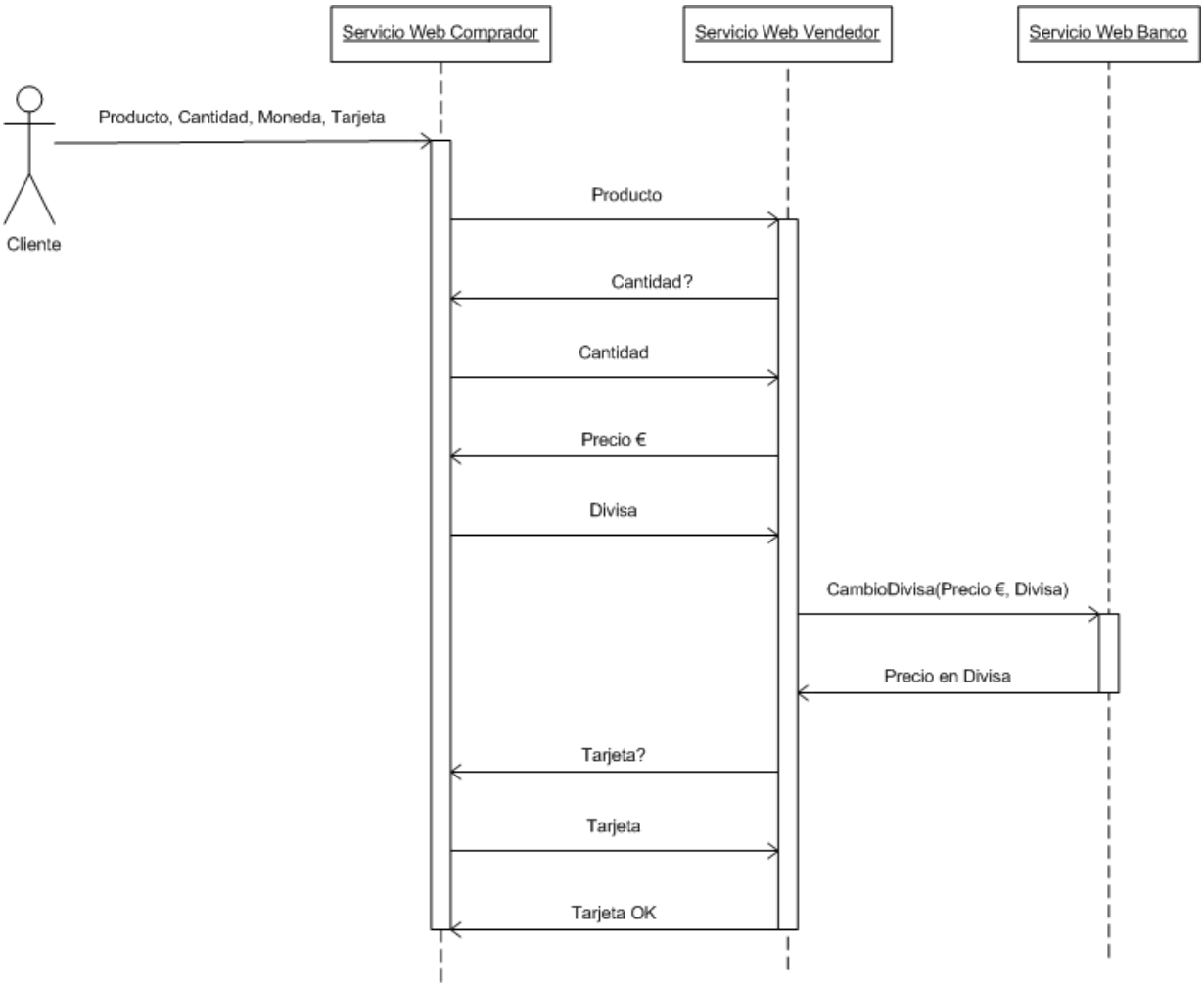
4.2.5 Modelado de BPEL4WS y WSMO

Para comenzar, se detalla un escenario en el que se propone el intercambio de datos entre tres servicios Web, que son los siguientes: un comprador, un vendedor y el banco. El servicio Web comprador envía al servicio Web vendedor las peticiones de su compra y el servicio Web vendedor tendrá que procesarlas. El servicio Web comprador en caso de que se haya solicitado pagar en una divisa diferente tendrá que acudir al servicio Web del banco para realizar el cambio de divisas.

Una vez propuesto el escenario se pasa a modelarlo en BPEL4WS y en WSMO. Para el primer caso, BPEL4WS, además de la explicación del modelado se adjuntan una serie de gráficos y el código propuesto para cada uno de los servicios Web estudiados. En WSMO se ofrece una explicación y el propio modelado.

4.2.5.1 Escenario para modelar

A continuación se presenta un diagrama en el que se pueden observar las secuencias de intercambio de datos entre los servicios:



El escenario propuesto para llevar a cabo el modelado, consta de tres servicios Web, el primero realiza labores de comprador, otro ejecuta tareas de vendedor y por último, el servicio Web del banco cuyo objetivo radica en realizar un cambio de divisas.

Para la interpretación de este diagrama, comenzamos a partir de un cliente que proporciona a un servicio Web llamado comprador los datos de una petición y que constan de un determinado producto, una cantidad de producto, una divisa en la que desea pagar su futura compra y los datos de la tarjeta de crédito, por ejemplo ProductoA, 1 unidad, libra, (tarjeta de crédito) xxxxxxxxxxxxxxx

En consecuencia, se producen los siguientes movimientos:

- El servicio Web comprador se pone en contacto para transmitir el producto deseado al servidor Web vendedor.
- El servicio Web vendedor solicita la cantidad de productos que desea el servicio Web comprador.
- El servicio Web comprador le envía al servicio Web vendedor la cantidad deseada.
- El servicio Web vendedor proporciona al servicio Web comprador el importe en euros de su compra.
- El servicio Web comprador envía al servicio Web vendedor la divisa en la que desea pagar.
- El servicio Web vendedor tiene que realizar una conversión mediante el precio y la divisa, para ello se pone en contacto con el servicio Web banco, al que le envía el precio de la compra y la divisa.
- El servicio Web banco devuelve al servicio Web vendedor la cantidad que tiene que cobrar en la divisa deseada.
- El servicio Web vendedor solicita al servicio Web comprador la tarjeta de crédito para abonar la compra.
- El servicio Web comprador envía los datos de la tarjeta de crédito al servicio Web vendedor.
- El servicio Web vendedor envía al servicio Web comprador la confirmación de la compra por su tarjeta de crédito.

De esta manera los tres servicios Web se interaccionarán con el objetivo de llevar a cabo una compra de manera exitosa.

4.2.5.2 Modelado de BPEL4WS

En la realización del modelado en BPEL4WS se han tenido en cuenta esencialmente los tres servidores Web que componen el escenario. Debido a ello, los servicios se modelan por separado.

De modo que, se modelará el servicio Web comprador, posteriormente el servicio Web vendedor y por último el servicio Web banco.

4.2.5.2.a Servicio Web Comprador

Comprador en BPEL4WS está compuesto por unos partner links referentes a los compañeros que tiene a su lado y que le proporcionan los datos necesarios en desarrollo.

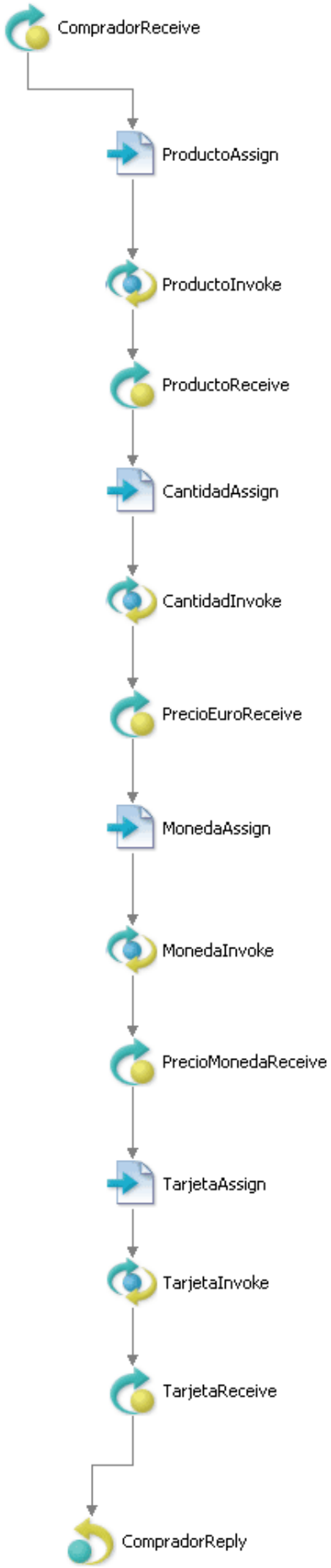
Las variables que se tienen en cuenta son según los datos que se intercambian, es decir producto, cantidad, moneda, precio y tarjeta.

La secuencia se puede resumir de la siguiente manera:

- El comprador recibe los datos a través de receive sobre los cuales va a trabajar.
- Asigna el valor correspondiente al producto e invoca la petición del producto.
- Recibe la petición de cantidad
- Asigna el valor de cantidad y los invoca.
- Recibe el precio en euros
- Asigna el valor de la divisa y la invoca.
- Asigna la tarjeta de crédito y la invoca
- Recibe la aceptación del envío de la tarjeta.

En su modelado también se definen los links entre los elementos.

Se adjunta el gráfico a continuación para entender de manera visual el modelado de comprador en BPEL4WS:


CompradorReceive
ProductoAssign
ProductoInvoke
ProductoReceive
CantidadAssign
CantidadInvoke
PrecioEuroReceive
MonedaAssign
MonedaInvoke
PrecioMonedaReceive
TarjetaAssign
TarjetaInvoke
TarjetaReceive
CompradorReply

A través del siguiente código se muestra detalladamente el modelado de comprador:

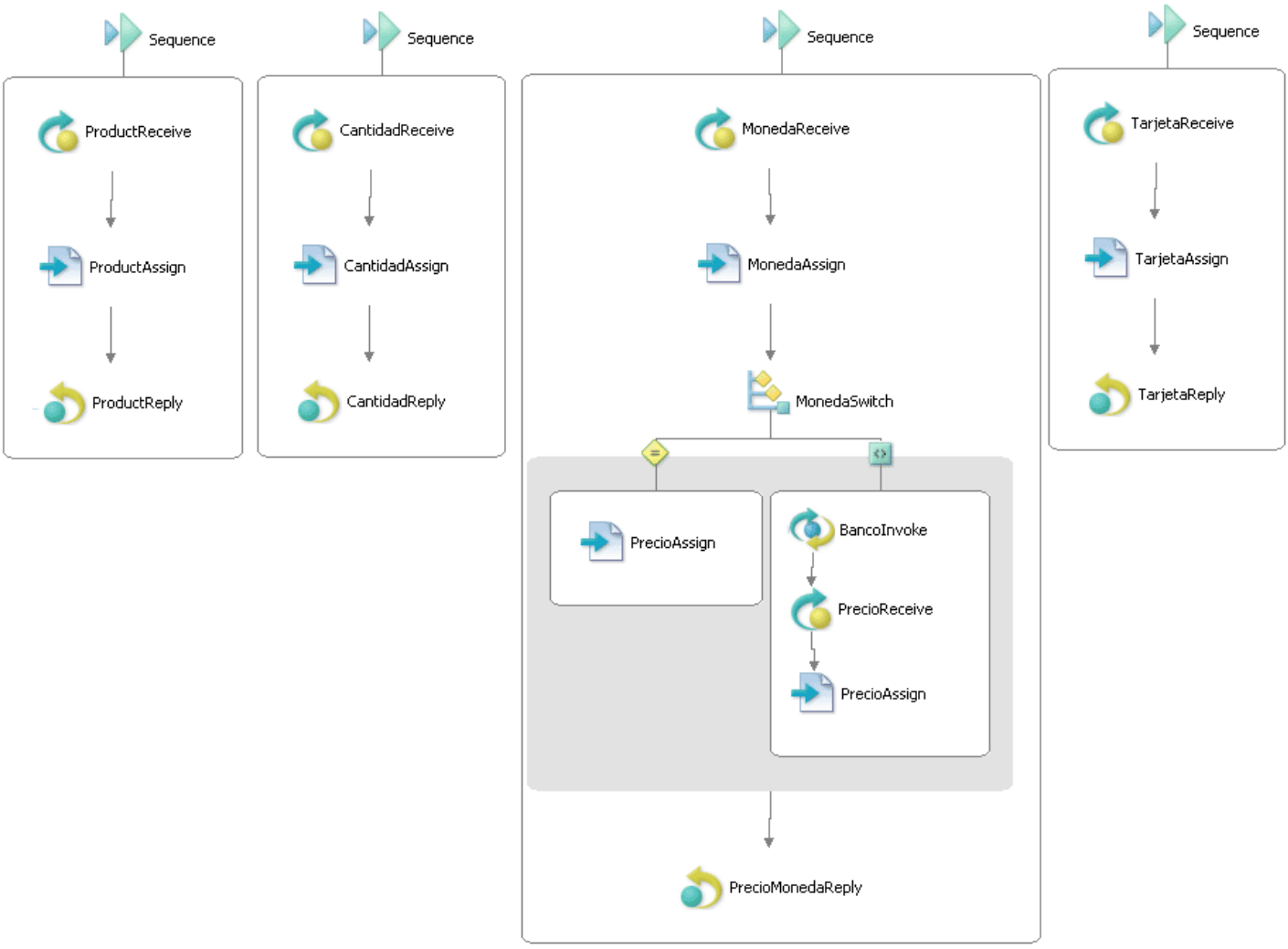
```
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="comprador"
suppressJoinFailure="yes" targetNamespace="http://comprador">
   <partnerLinks>
      <partnerLink name="cliente"/>
      <partnerLink name="vendedor"/>
   </partnerLinks>
   <partners>
      <partner name="P1">
         <partnerLink name="cliente"/>
         <partnerLink name="vendedor"/>
      </partner>
   </partners>
   <variables>
      <variable name="Producto" type="xsd:integer"/>
      <variable name="Cantidad" type="xsd:integer"/>
      <variable name="Moneda" type="xsd:string"/>
      <variable name="Tarjeta" type="xsd:long"/>
      <variable name="ProductoB" type="xsd:integer"/>
      <variable name="CantidadB" type="xsd:integer"/>
      <variable name="MonedaB" type="xsd:string"/>
      <variable name="TarjetaB" type="xsd:long"/>

   </variables>
   <flow>
      <links>
         <link name="L1"/>
         <link name="L2"/>
         <link name="L3"/>
         <link name="L4"/>
         <link name="L5"/>
         <link name="L6"/>
         <link name="L8"/>
         <link name="L7"/>
         <link name="L9"/>
         <link name="L10"/>
         <link name="L11"/>
         <link name="L12"/>
         <link name="L13"/>
      </links>
      <receive createInstance="yes"  name="CompradorReceive"
partnerLink="cliente" operation="CompradorReceive" variable="Producto">
         <source linkName="L1"/>
      </receive>
      <assign name="ProductoAssign">
         <target linkName="L1"/>
         <source linkName="L2"/>
         <copy>
            <from endpointReference="partnerRole" partnerLink="cliente"/>
            <to variable="Producto"/>
         </copy>
      </assign>
      <invoke name="ProductoInvoke" partnerLink="vendedor"
inputVariable="Producto" operation="ProductoInvoke"
outputVariable="ProductoB">
         <target linkName="L2"/>
```

```
         <source linkName="L3"/>
      </invoke>
      <receive name="ProductReceive" createInstance="yes"
partnerLink="vendedor" operation="ProductReceive" variable="Producto">
         <target linkName="L3"/>
         <source linkName="L4"/>
      </receive>
      <assign name="CantidadAssign">
         <target linkName="L4"/>
         <source linkName="L5"/>
         <copy>
            <from endpointReference="partnerRole" partnerLink="cliente"/>
            <to variable="Cantidad"/>
         </copy>
      </assign>
      <invoke name="CantidadInvoke" partnerLink="vendedor"
inputVariable="Cantidad" operation="CantidadInvoke"
outputVariable="CantidadB">
         <target linkName="L5"/>
         <source linkName="L6"/>
      </invoke>
      <assign name="MonedaAssign">
         <target linkName="L7"/>
         <source linkName="L8"/>
         <copy>
            <from endpointReference="myRole" partnerLink="cliente"/>
            <to variable="Moneda"/>
         </copy>
      </assign>
      <receive name="PrecioEuroReceive" createInstance="yes"
partnerLink="vendedor" operation="PrecioEuroReceive" variable="Precio">
         <target linkName="L6"/>
         <source linkName="L7"/>
      </receive>
      <invoke name="MonedaInvoke" partnerLink="vendedor"
inputVariable="Moneda" operation="MonedaInvoke" outputVariable="MonedaB">
         <target linkName="L8"/>
         <source linkName="L9"/>
      </invoke>
      <reply name="CompradorReply" operation="CompradorReply"
partnerLink="cliente" >
         <target linkName="L13"/>
      </reply>
      <receive name="PrecioMonedaReceive" createInstance="yes"
partnerLink="vendedor" operation="PrecioMonedaReceive" variable="Precio">
         <target linkName="L9"/>
         <source linkName="L10"/>
      </receive>
      <assign name="TarjetaAssign">
         <target linkName="L10"/>
         <source linkName="L11"/>
         <copy>
            <from endpointReference="myRole" partnerLink="cliente"/>
            <to variable="Tarjeta"/>
         </copy>
      </assign>
      <invoke name="TarjetaInvoke" partnerLink="vendedor"
inputVariable="Tarjeta" operation="TarjetaInvoke" outputVariable="TarjetaB">>
         <target linkName="L11"/>
         <source linkName="L12"/>
```

```
      </invoke>
      <receive name="TarjetaReceive" createInstance="yes"
name="TarjetaReceive" partnerLink="vendedor" operation="TarjetaReceive"
variable="Tarjeta">
         <target linkName="L12"/>
         <source linkName="L13"/>
      </receive>
   </flow>
</process>
```

4.2.5.2.b Servicio Web Vendedor

Para el servicio Web del vendedor se detalla el modelado acorde al siguiente gráfico:



En el gráfico se puede comprobar como existen cuatro secuencias paralelas que se ejecutan según sean necesarias en el flujo de datos.

La primera de las secuencias se ejecuta cuando llega un producto al servicio Web vendedor, que se asigna y se devuelve al servicio Web que realiza la petición, en este caso el comprador, preguntando en este caso por la cantidad de productos requeridos.

La segunda secuencia se produce al llegar al servicio Web vendedor la cantidad requerida, que se asigna y se devuelve preguntando por la moneda al mismo tiempo que se muestra el precio en euros.

La tercera secuencia sucede cuando llega una moneda al servicio Web vendedor, en ella se analiza si la divisa llegada es igual a euros o diferente, si es diferente, se produce una llamada al servicio Web banco para realizar un cambio de divisas, por el cual se devuelve el precio de la compra en la nueva divisa.

En la cuarta y última secuencia, se recibe los datos de la tarjeta que se asignan y se devuelve la tarjeta finalmente aceptada.

Se realizan los partner links con los servicios Web que se relaciona directamente el vendedor, es decir con el comprador y con el banco.

Al igual que en el servicio Web comprador, se definen las variables referentes al precio, producto, cantidad, tarjeta y moneda.

A continuación se muestra el código para concretar el modelado del servicio Web vendedor.

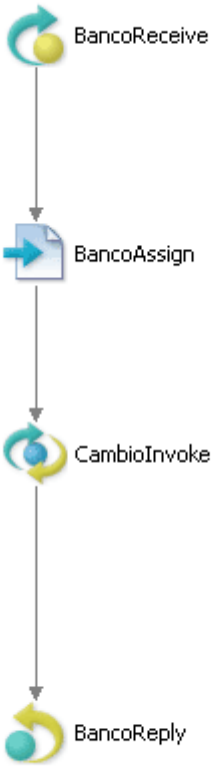
```
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="vendedor"
suppressJoinFailure="yes" targetNamespace="http://vendedor">
   <partnerLinks>
      <partnerLink name="comprador"/>
      <partnerLink name="banco"/>
   </partnerLinks>
   <variables>
      <variable name="Producto" type="xsd:integer"/>
      <variable name="Cantidad" type="xsd:integer"/>
      <variable name="Precio" type="xsd:double"/>
      <variable name="PrecioB" type="xsd:double"/>
      <variable name="Tarjeta" type="xsd:long"/>
      <variable name="Moneda" type="xsd:string"/>
   </variables>
   <flow>
      <sequence>
         <receive name="CantidadReceive" partnerLink="comprador"
createInstance="yes" operation="CantidadReceive" variable="Cantidad"/>
         <assign name="CantidadAssign">
            <copy>
               <from endpointReference="myRole" partnerLink="comprador"/>
               <to variable="Cantidad"/>
```

```
            </copy>
         </assign>
         <reply name="CantidadReply" partnerLink="comprador"
operation="CantidadReply" variable="Cantidad" />
      </sequence>
      <sequence>
         <receive name="ProductReceive" partnerLink="comprador"
createInstance="yes" operation="ProductReceive" variable="Producto"/>
         <assign name="ProductAssign">
            <copy>
               <from endpointReference="myRole" partnerLink="comprador"/>
               <to variable="Producto"/>
            </copy>
         </assign>
         <reply name="ProductReply" partnerLink="comprador"
operation="ProductReply" variable="Producto" />
      </sequence>
      <sequence>
         <receive name="MonedaReceive" partnerLink="comprador"
createInstance="yes" operation="MonedaReceive" variable="Moneda"/>
         <assign name="MonedaAssign">
            <copy>
               <from endpointReference="myRole" partnerLink="comprador"/>
               <to variable="Moneda"/>
            </copy>
         </assign>
         <switch name="MonedaSwitch">
            <case condition="bpws:getVariableData('Moneda') =Euro">
               <assign name="PrecioAssign">
                  <copy>
                     <from variable="Precio"/>
                     <to variable="Precio"/>
                  </copy>
               </assign>
            </case>
            <otherwise>
               <flow>
                  <links>
                     <link name="L3"/>
                     <link name="L4"/>
                  </links>
                  <assign name="PrecioAssign">
                     <target linkName="L4"/>
                     <copy>
                        <from endpointReference="myRole"
partnerLink="banco"/>
                        <to variable="Precio"/>
                     </copy>
                  </assign>
                  <invoke name="BancoInvoke" partnerLink="banco"
inputVariable="Precio" operation="BancoInvoke" outputVariable="PrecioB">
                     <source linkName="L3"/>
                  </invoke>
                  <receive name="PrecioReceive" partnerLink="banco"
createInstance="yes" operation="PrecioReceive" variable="Precio">
                     <target linkName="L3"/>
                     <source linkName="L4"/>
                  </receive>
               </flow>
            </otherwise>
```

```
         </switch>
         <reply name="PrecioMonedaReply" partnerLink="comprador"
operation="PrecioMonedaReply" variable="PrecioB"/>
      </sequence>
      <sequence>
         <receive name="TarjetaReceive" partnerLink="comprador"
createInstance="yes" operation="TarjetaReceive" variable="Tarjeta"/>
         <assign name="TarjetaAssign">
            <copy>
               <from endpointReference="myRole" partnerLink="comprador"/>
               <to variable="Tarjeta"/>
            </copy>
         </assign>
         <reply name="TarjetaReply" partnerLink="comprador"
operation="TarjetaReply" variable="Tarjeta" />
      </sequence>
   </flow>
</process>
```

4.2.5.2.c Servicio Web banco

Para el servicio Web del banco se muestra el modelado en el siguiente gráfico



En este servicio el banco recibe del servicio Web comprador un precio y una divisa y

posteriormente tiene que realizar la operación que permita realizar el cambio según el valor de la divisa en el momento en el que se produzca la petición.
Por último devuelve la información actualizada con el nuevo precio al servicio que lo seleccionó, en este caso el servicio Web comprador.

Se muestra el código con el detalle del modelado del servicio Web del banco.

```
<process xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" name="banco"
suppressJoinFailure="yes" targetNamespace="http://banco">
   <partnerLinks>
      <partnerLink name="Vendedor"/>
      <partnerLink name="Cambio"/>
   </partnerLinks>
   <variables>
      <variable name="Precio" type="xsd:double"/>
      <variable name="PrecioB" type="xsd:double"/>
      <variable name="Moneda" type="xsd:string"/>
   </variables>
   <flow>
      <links>
         <link name="L1"/>
         <link name="L2"/>
         <link name="L3"/>
      </links>
      <receive name="BancoReceive" partnerLink="Vendedor"
createInstance="yes"  operation="BancoReceive" variable="Precio">
         <source linkName="L1"/>
      </receive>
      <reply name="BancoReply" partnerLink="Vendedor" operation="BancoReply"
variable="PrecioB">
         <target linkName="L3"/>
      </reply>
      <assign name="BancoAssign">
         <target linkName="L1"/>
         <source linkName="L2"/>
         <copy>
            <from endpointReference="myRole" partnerLink="Vendedor"/>
            <to variable="Moneda"/>
         </copy>
         <copy>
            <from endpointReference="myRole" partnerLink="Vendedor"/>
            <to variable="Precio"/>
         </copy>
      </assign>
      <invoke name="CambioInvoke" partnerLink="Cambio" inputVariable="Precio"
operation="CambioInvoke" outputVariable="PrecioB">
         <target linkName="L2"/>
         <source linkName="L3"/>
      </invoke>
   </flow>
</process>
```

4.2.5.3 Modelado de WSMO

Para llevar a cabo el modelado en WSMO he seguido los puntos que aportan su documentación, basándome en las ontologías, los mediadores, las metas y los servicios Web.

Para comenzar, he definido una ontología principal que es compraProducto y esta importa otras ontologías necesarias como son la ontologíaCantidades, la ontologíaMonedas y la ontologíaTarjetas.

Se define el concepto producto y posteriormente las relaciones, es uno de los puntos más complejos ya que engloba las referencias de varios intercambios de datos. Se hacen las siguientes relaciones:

- producto: para solicitar un producto
- solicitaCantidad: cuando se solicita una cantidad
- cantidadProducto: asigna una cantidad a un producto
- monedaProducto: asigna una divisa a un producto
- precioProducto: asigna un precio a un producto
- cambioDivisa: función para cambiar un precio de divisa
- solicitaTarjeta: cuando solicita una tarjeta.
- compruebaTarjeta: al evaluar los datos de la tarjeta
- tarjetaOK: relaciona al confirmar la tarjeta.

Se define la meta, donde se especifica que en este caso se desea comprar un producto en la divisa en libras y se aporta la tarjeta.

Se definen los mediadores, en este caso dos tipos de mediadores, OO Mediators para resolver entre ontologías y WW Mediators, para resolver entre servicios Web.

Los OO Mediators definidos, son para resolver entre las siguientes ontologías:

- ContidadProducto
- TarjetaProducto

- MonedaProducto

Los WW Mediators definidos son entre los servicios Web propuestos, es decir:

- CompradorVendedor
- VendedorBanco

Para finalizar se defienen los servicios Web, en este caso son tres; comprador, vendedor y banco, y donde se definen de cada uno de ellos las precondiciones, postcondiciones, assumptions y efectos.

El modelado se puede observar de manera detallada en las siguientes líneas:

```
namespace {_"http://compraProducto",
          dc     _"http://purl.org/dc/elements/1.1#",
          can    _"http://ontologiaCantidades#",
          tar     _"http://ontologiaTarjetas#",
          mon    _"http://ontologiaMonedas#",
          foaf   _"http://xmlns.com/foaf/0.1/",
           wsml   _"http://www.wsmo.org/wsml/wsml-syntax#"
          }

ontology  _"http://compraProducto"
  nonFunctionalProperties
                  dc#title hasValue "Compra de producto"
                  dc#identifier hasValue _"http://compraProducto"
                  dc#description hasValue  "Ontología para la compra de
producto"
                  dc#date hasValue _date(2006,12,06)
                  dc#format hasValue "text/x-wsml"
                  dc#language hasValue "es-ES"
                  wsml#version hasValue "$Revision 0.1$"
  endNonFunctionalProperties

  importsOntology  {
          _"http://ontologíaCantidades",
                     _"http://ontologíaTarjetas",
                   _"http://ontologiaMoneda}

  concept producto
          nombre ofType _string
          precio ofType _double


       nonFunctionalProperties
          dc#description hasValue "representa al producto seleccionado"
       endNonFunctionalProperties
          productoTarjeta impliesType tar#numero
          productoCantidad impliesType can#numero
              productoMoneda impliesType mon#divisa
```

```
relation Producto(ofType producto)
        nonFunctionalProperties
                  dc#description hasValue "Solicita un producto"
                   dc#relation hasValue {AsignaProducto}
        endNonFunctionalProperties


   axiom AsignaProducto
              definedBy
                  Produto(?x)


relation SolicitaCantidad(ofType tarjeta)
        nonFunctionalProperties
                  dc#description hasValue "Solicitar cantidad"
                  dc#relation hasValue {SolicitarCantidad}
        endNonFunctionalProperties


   axiom SolicitarCantidad
              definedBy
                  SolicitarCantidad (?x)


relation cantidadProducto(ofType can#numero, ofType producto)
        nonFunctionalProperties
                  dc#description hasValue "Asigna una cantidad a un
producto"
                   dc#relation hasValue {AsignaCantidadProducto}
        endNonFunctionalProperties


   axiom AsignaCantidadProducto
              definedBy
                  cantidadProduto(?x,?y)

relation monedaProducto(ofType mon#divisa, ofType producto)
        nonFunctionalProperties
                  dc#description hasValue "Asigna una moneda a la
compra de un producto"
                   dc#relation hasValue {AsignaMoneda}
        endNonFunctionalProperties


   axiom AsignaMoneda
              definedBy
                  MonedaProduto(?x,?y)


relation precioProducto(ofType precio, ofType producto)
        nonFunctionalProperties
                  dc#description hasValue "Asigna un precio a un
producto"
                  dc#relation hasValue {AsignaPrecio}
        endNonFunctionalProperties
```

```
  axiom AsignaPrecio
              definedBy
                  PrecioProduto(?x,?y)


relation SolicitaTarjeta (ofType tarjeta)
       nonFunctionalProperties
                  dc#description hasValue "Solicita la tarjeta"
                  dc#relation hasValue {SolicitaTarjeta}
       endNonFunctionalProperties


  axiom SolicitaTarjeta
              definedBy
                  SolicitarTarjeta (?x)


relation CompruebaTarjeta (ofType tarjeta)
       nonFunctionalProperties
                  dc#description hasValue "Comprueba la tarjeta"
                  dc#relation hasValue {CompruebaTarjeta}
       endNonFunctionalProperties


  axiom CompruebaTarjeta
              definedBy
                  ComprobarTarjeta (?x)


relation tarjetaOk(ofType tarjeta)
       nonFunctionalProperties
                  dc#description hasValue "Tarjeta validada"
                  dc#relation hasValue {TarjetaValida}
       endNonFunctionalProperties


  axiom TarjetaValida
              definedBy
                  TarjetaOk(?x)


relation CambioDivisa(ofType precio, ofType mon#divisa, ofType _double)

      nonFunctionalProperties
                  dc#description hasValue "Función que devuelve el
precio de producto en euros en la divisa requerida"
                  dc#relation hasValue {FuncionCambioDivisa}
            endNonFunctionalProperties


  axiom FuncionCambioDivisa
               definedBy
          cambio(?x,?y,?z)


namespace {_"http://compraProductoMetas#",
```

```
                    pro        _"http://compraProducto",


goal  _"http://ComprarProductosConLibras"

importsOntology {_" http://compraProducto" }

  capability
      postcondition
           definedBy
                     compraProducto[
                     pro#nombre hasValue “producto”,
                     can#numero hasValue “1”,
           mon#divisa hasValue “libra”,
           tar#numero hasValue “xxxxxxxxxxx”
            ]


   namespace{_"http://example.org/mediators#",
           dc     _"http://purl.org/dc/elements/1.1" ,
           wsml   _"http://www.wsmo.org/wsml/wsml-syntax#"
}


ooMediator _"http://cantidadProductoMediador"
  nonFunctionalProperties
                  dc#title hasValue "OO Mediator cantidad - producto"
                  dc#description hasValue “OO Mediador entre cantidad y
producto”
           dc#language hasValue "ES-es"
                  dc#relation hasValue {_"http://CompraProducto/",
_"http://ontologíaCantidades"}
  endNonFunctionalProperties
  source _" http://ontologíaCantidades"
  target _" http://CompraProducto "



ooMediator _"http://TarjetaProductoMediador"
  nonFunctionalProperties
                  dc#title hasValue "OO Mediator tarjeta - producto"
                  dc#description hasValue “OO Mediador entre tarjeta y
producto”
           dc#language hasValue "ES-es"
                  dc#relation hasValue {_"http://CompraProducto/",
                       _"http://ontologíaTarjetas"}
  endNonFunctionalProperties
  source _" http://ontologíaTarjetas"
  target _" http://CompraProducto "



ooMediator _"http://MonedaProductoMediador"
  nonFunctionalProperties
                  dc#title hasValue "OO Mediator mMoneda - producto"
                  dc#description hasValue “OO Mediador entre moneda y
producto”
```

```
                    dc#language hasValue "ES-es"
                    dc#relation hasValue {_"http://CompraProducto/",
                         _"http://ontologíaMonedas"}

   endNonFunctionalProperties
  source _" http://ontologíaMonedas"
  target _" http://CompraProducto "



wwMediator _"http://CompradorVendedorMediador"
    nonFunctionalProperties
                    dc#title hasValue "WWMediador entre Comprador y
Vendedor"
  endNonFunctionalProperties
  source  _"http://comprador"
 target _"http://vendedor"



wwMediator _"http://VendedorBancoMediador"
    nonFunctionalProperties
                    dc#title hasValue "WWMediador entre Vendedor y Banco"
  endNonFunctionalProperties
  source  _"http://vendedor"
 target _"http://banco"



namespace {_"http://comprador#",
                    dc       _"http://purl.org/dc/elements/1.1#",
                    foaf     _"http://xmlns.com/foaf/0.1/",
                    wsml   _"http://www.wsmo.org/wsml/wsml-syntax#",
                    pro  _"http://compraProducto"}

webService _"http://comprador"


importsOntology _"http://comprarProducto"


capability compraCapability

   precondition
      nonFunctionalProperties
                    dc#description hasValue "Solicita la compra de una
cantidad de producto, pagada en moneda en una divisa determinada a
través de una tarjeta."
      endNonFunctionalProperties
      definedBy
          ?comprarProducto[
          productoTarjeta hasValue ?numero
          productoCantidad hasValue ?numero
          productoMoneda hasValue ?divisa
                  ] memberOf pro#rproducto


  assumption
```

```
      nonFunctionalProperties
          dc#description hasValue "La divisa con el precio del producto
está en euros y tiene que presentarse en la divisa definida por la
moneda."
      endNonFunctionalProperties
      definedBy
                  {PrecioProducto(?precio)}


  postcondition
      nonFunctionalProperties
          dc#description hasValue "Se realiza el pago con la tarjeta."
      endNonFunctionalProperties
      definedBy
          {CompruebaTarjeta(?tarjeta)}


  effect
      nonFunctionalProperties
          dc#description hasValue "Se realiza la compra de un
producto."
      endNonFunctionalProperties
      definedBy
          CompruebaTarjeta(?tarjeta)



namespace {_"http://vendedor#",
                  dc       _"http://purl.org/dc/elements/1.1#",
                  foaf     _"http://xmlns.com/foaf/0.1/",
                  wsml   _"http://www.wsmo.org/wsml/wsml-syntax#",
                  pro  _"http://compraProducto"}

webService _"http://vendedor"


importsOntology _"http://comprarProducto"


capability ventaCapability

   precondition
      nonFunctionalProperties
                  dc#description hasValue "Proporciona los datos
necesarios para efectuar la venta."
      endNonFunctionalProperties
      definedBy
          {Producto(?producto),
          CantidadProducto(?cantidad),
          MonedaProducto(?moneda)
          CompruebaTarjeta(?tarjeta) }



  assumption
      nonFunctionalProperties
          dc#description hasValue "Se tiene que solicitar los datos
para efectuar la venta"
      endNonFunctionalProperties
```

```
        definedBy
            {SolicitaCantidad(?cantidad),
                    MonedaProducto(?moneda),
            SolicitaTarjeta(?tarjeta)}


    postcondition
        nonFunctionalProperties
            dc#description hasValue "Se acepta la tarjeta."
        endNonFunctionalProperties
        definedBy
            {tarjetaOk(?tarjeta)}


    effect
        nonFunctionalProperties
            dc#description hasValue "Se realiza la venta de un producto."
        endNonFunctionalProperties
        definedBy
            {tarjetaOk(?tarjeta)}



namespace {_"http://banco#",
                    dc       _"http://purl.org/dc/elements/1.1#",
                    foaf     _"http://xmlns.com/foaf/0.1/",
                    wsml   _"http://www.wsmo.org/wsml/wsml-syntax#",
                    pro  _"http://compraProducto"}

webService _"http://banco"


importsOntology _"http://comprarProducto"


capability bancoCapability

   precondition
      nonFunctionalProperties
                    dc#description hasValue "Se proporciona el precio y
la divisa."
      endNonFunctionalProperties
      definedBy
          {PrecioProducto(?precio),
        SolicitaMoneda(?moneda) }



    assumption
        nonFunctionalProperties
            dc#description hasValue "Se realiza la conversión a la divisa
solicitada."
        endNonFunctionalProperties
        definedBy
                    cambioDivisa(pro#precio,pro#divisa,?precio)


    postcondition
        nonFunctionalProperties
```

```
            dc#description hasValue "Se devuelve el nuevo precio."
        endNonFunctionalProperties
        definedBy
            {PrecioProducto(?precio)}


    effect
        nonFunctionalProperties
            dc#description hasValue "Se realiza la conversión de la
divisa devolviendo el nuevo precio"
        endNonFunctionalProperties
        definedBy
       {PrecioProducto(?precio)}
```

4.3 Conclusiones

He conocido, como conclusiones, algunas especificaciones que acercan más el BPEL4WS a WSMO, debido a que en sus orígenes no tienen unas características similares. Siendo esto por causa de la parte que abarcan del total de la envergadura del problema inicial. Por ejemplo a través de METEOR-S, desarrollado por la universidad de Georgia, se puede realizar una comparación más exacta por sus especificaciones. Incluso aporta una herramienta de diseño de procesos (PDT) para facilitar el desarrollo de procesos de negocios complejos.

Siguiendo con las ideas aportadas en la comparativa anterior, existen diferencias importantes para llevar a cabo el modelado de un escenario dependiendo si se realiza en BPEL4WS o en WSMO, quizás las más importantes vienen dadas por la madurez que ha conseguido en los últimos años BPEL4WS, ya que para realizar este modelado existen diversas herramientas ya sean libres o de pago y por supuesto abundante documentación. En el caso de WSMO no se puede decir lo mismo, ya que actualmente está en fase de menor madurez y es difícil obtener información aparte de la documentación oficial.

4.4 Líneas de desarrollo futuro

Como línea de futuro, después de las comparaciones técnicas he observado tendencias hacia la “semanticalización” de BPEL4WS, existiendo algunas extensiones que tratan desde hace tiempo el problema de la carencia de semántica, por ejemplo BPWS4J.

Una consecuencia, al margen de las propias tecnologías en si mismas, viene dada por el uso de servicios Web, y en especial de los servicios Web semánticos. Puede desencadenarse en un cambio en los escenarios de uso de los programas informáticos, donde mientras ahora se ejecutan programas instalados en el propio cliente, puede tender hacia el uso de los mismos a través de servicios desde servidores. Este aspecto puede tener diversas consecuencias, si bien pueden enriquecerse los servicios a los que se puede acceder, también puede aprovecharse para realizar servicios de pago en contraposición a la filosofía de la Web como red libre y abierta a todos.

5. Glosario

BPEL4WS: (Business Process Execution Language for Web Services) en castellano, Lenguaje de ejecución de procesos de negocios para servicios Web, es un lenguaje basado en XML cuyo objetivo principal se basa en la orequestación de servicios Web.

SOA: (Service-Oriented Architecture) en castellano, Arquitectura orientada a servicios, define la utilización de servicios para conseguir los objetivos requeridos por un software, proporcionando la metodología y un marco de trabajo.

SOAP: (Simple Object Access Protocol), protocolo que define la manera de conectar dos objetos para intercambiar mensajes.

SSOA: (Semantic Service-Oriented Architecture) la versión para servicios semánticos de SOA

UDDI: (Universal Description, Discovery, and Integration) servicio de directorios para servicios Web basado en XML

URI: (Uniform Resource Identifier) en castellano, Identificador uniforme de recursos, es un identificador único para la localización de un recurso.

WS-BPEL: (Web Services Business Process Execution Language) ver BPEL4WS.

WS-I: (Web Services Interoperability) en castellano, Interoperabilidad de los servicios Web, tiene la intención integradora de favorecer el principio de la interoperabilidad entre servicios Web de una manera estructurada y coherente.

WSDL: (Web Services Description Language) estándar para la comunicación entre servicios Web.

WSML: (Web Service Modeling Language) en castellano, Lenguaje modelo de servicios Web, es la lengua en la que se basa WSMO ya que requiere que una lengua formal que escriba las anotaciones de servicios de Web según el modelo conceptual presentado por WSMO.

WSMO: (Web Service Modeling Ontology) en castellano, Ontología de modelado para servicios Web, define las interfaces de orquestación y coreografía para describir la interacción entre servicios a través de metas, ontologías, mediadores y capacidades.

WSMX: (Web Service Modelling eXecution environment) en castellano, Entorno de ejecución y modelado de servicios Web, tiene como objetivo el desarrollo de un entorno de ejecución para el descubrimiento dinámico, la selección, la mediación, la invocación

y la interoperación de servicios Web semánticos basados en la especificación WSMO. WSMX proporciona una arquitectura de referencia y la puesta en práctica.

XML: (eXtensible Markup Language) en castellano, lenguaje de marcas extensible y es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C), es un lenguaje particular dado que se usa como estándar en el intercambio de información de manera estructurada entre distintas plataformas de manera concisa, fácil y extensible.

XML Schema: es un tipo de XML que cumple el estándar propuesto y que se utiliza para la definición de tipos de documentos o de datos para especificar y ofrecer el contenido de cualquier documento de manera que facilita su clasificación. Schema significa esquema.

XPath: (XML Path Language) es un lenguaje cuya sintaxis permite buscar y seleccionar partes jerarquicas de un documento XML. No es un lenguaje XML.

6. Bibliografía

Enlaces de Internet

- **http://www.w3.org**
  Sitio referente de la Web semántica y otros aspectos de la Web

- **http://www-128.ibm.com/developerworks/library/specification/ws-bpel/**
  Sitio Web acerca del Lenguaje de Ejecución de Procesos de Negocio.

- **http://www.wsmo.org**
  Sitio Web del modelo conceptual para Servicios Web

- **http://www.semanticWeb.org**
  Sitio Web sobre Web semántica.

- **http://www.infraWebs-eu.org**
  Sitio Web europeo acerca de Servicios Web

- **http://www.oasis-open.org**
  Sitio Web acerca del comité OASIS

- **http://www.uddi.org**
  Sitio Web acerca de UDDI.

- **http://www.active-endpoints.com**
  Sitio Web de ActiveBPEL Designer.