Universidad de las Ciencias Informáticas

Facultad 7

Trabajo de Diploma para optar por el Título de Ingeniero en Ciencias Informáticas

Título: Proceso de migración de los Servicios Web del Sistema de Información para la Salud

Autores: **Yurismaury Aguila González**
Jayler Amaró Izquierdo

Tutor: **Ing. Yoenny Pérez Romero**

Ciudad de La Habana 2008
"Año 50 de la Revolución"

DECLARACIÓN DE AUTORÍA

Nosotros, Yurismaury Aguila González y Jayler Amaró Izquierdo, declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmamos la presente a los 9 días del mes de Julio del año 2008.

Yurismaury Aguila González

_____
Firma del Autor

Jayler Amaró Izquierdo

_____
Firma del Autor

Yoenny Pérez Romero

_____
Firma del Tutor

Datos de contacto:

Ing. Yoenny Pérez Romero *(yoenny@uci.cu):* Graduado como Ingeniero Informático en el Instituto Superior Politécnico José Antonio Echeverría en el 2005. Posee 3 años de experiencia laboral, se ha desempeñado como Profesor Instructor en la Universidad de las Ciencias Informáticas (UCI), vinculado siempre a las asignaturas de la especialidad. Durante este período ha estado desarrollando conjuntamente con un equipo de trabajo de la UCI y la empresa SOFTEL una solución informática que automatiza y gestiona los procesos inherentes a la Atención Primaria de la Salud en Cuba (APS), donde ha fungido como Líder de Proyecto. Actualmente cursa la parte lectiva de la Maestría Informática Aplicada que se desarrolla en la UCI.

“...la verdad es que da vergüenza ver algo y no aprenderlo, el hombre no ha de descansar, hasta que aprenda todo lo que ve.”

José Martí

AGRADECIMIENTOS

A mis padres por guiarme siempre en el camino correcto de la vida, brindarme su apoyo en todo momento y que gracias a ellos, me he convertido en el joven que soy.
A mi hermano por su apoyo y cariño, por ser simplemente él.
A mi familia por su preocupación constante.
A Aily, por tanto cariño, amor, ternura, apoyo y sus consejos en todo momento.
A mi tutor, por el tiempo brindado, sus consejos y ayuda a lo largo del desarrollo del trabajo.
A mis compañeros de la universidad por brindarme su amistad a lo largo de estos 5 años.
A mis amigos, por todo el apoyo y ayuda brindada.
A mis vecinos por su preocupación y solidaridad.
A la Universidad por las alegrías y experiencias, adquiridas en estos 5 años.
A la Revolución y a nuestro Comandante en Jefe Fidel Castro.

Yurismaury

Primeramente quisiera agradecer a mis adorables padres, no se que haría sin ellos, me supieron guiar, me supieron dar la fuerza necesaria en todo momento, me hicieron andar por el buen camino.
A mi abuela (mi segunda madre) por sus sabios consejos gracias a su basta experiencia. Gracias abuela por todo tu amor.
A mi hermano y su novia por también apoyarme mucho y estar pendientes de cuanto me pasaba siempre.
A mí querida chiquita (mi novia Anisley), me dio fuerzas su amor, me motivó, influyó en mí de manera muy positiva y eso me hizo bien.
A mi tutor, por su ayuda, por sus consejos, por sus críticas constructivas que fueron demasiado útiles en todo momento.
A todos mis compañeros de grupo y a mis amigos que me apoyaron en todo momento. A mis más allegados vecinos, desde lejos siempre preocupados.
A mi compañero de tesis, por su ayuda durante estos 5 años.
A la Revolución, a nuestro Comandante en Jefe Fidel Castro Ruz por darme la posibilidad de cursar en una universidad de la calidad de la UCI.

Jayler

DEDICATORIA

… A mi familia, Aily, mis vecinos y amistades, por su apoyo incondicional en todo momento, a todos gracias…

Yurismaury

A mis tres grandes padres Elida, Pedro y Tomasa (abuela), mis amigos, mis compañeros, mi ejemplo, mi guía, mi vida.
A mi hermano Jancy y su compañera Dayimi.
A mi novia hermosa.
A todos mis amigos donde quiera que estén.
A todos muchas gracias.

Jayler

RESUMEN

El presente trabajo se realiza como propuesta de la empresa de Softel en conjunto con la Universidad de las Ciencias Informáticas para darle respuesta a la necesidad de mejorar de forma radical la interoperabilidad que presentan los servicios web aplicando la arquitectura SOA (Service-Oriented Architecture).

Para ello se ha enfocado en el estudio del XML y más específicamente en los WSDL, como una forma de acceder a estos y que es vital para la comunicación entre servicios. Los Servicios Web que servirán como bloques de construcción (Building-Blocks) para la arquitectura deben cumplir con un grupo de requisitos de calidad que aseguren su interoperabilidad, los cuales deben estar normados por la WS-I (Web Service Interoperability), por lo que el objetivo del trabajo es realizar una guía de pasos a seguir para adoptar la norma WS-I.

Con este trabajo se logrará obtener un diagnóstico de faltas en cuanto a la correspondencia de los Servicios Web de Softel con el estándar WS-I. Obtención de un catálogo de patrones de implementación de Servicios Web que cumplan con la norma WS-I y así poder estandarizar los Servicios Web de la empresa y de esta manera eliminar los problemas que existen en la actualidad referente a la comunicación.

ÍNDICE

ÍNDICE

ÍNDICE

INTRODUCCIÓN

La empresa de Softel, dedicada a satisfacer las necesidades de la salud cubana en lo que a servicios web respecta, necesita en la actualidad con un procedimiento que mejore aspecto en la interoperabilidad de estos servicios que realiza, mejorando así todo tipo de comunicación que estos puedan desarrollar.

Para la implementación de Servicios Web, la interoperabilidad es tal vez el principio más importante. Por ello en dicha empresa se quiere asimilar e implantar para su proceso de desarrollo de software una arquitectura orientada a servicios. Para esto se ha decidido asimilar tecnologías de la familia de SOA/BPM. Ésta evolución requiere que los Servicios Web, que servirán como bloques de construcción (Building-Blocks) para la nueva arquitectura, cumplan con un grupo de requisitos de calidad que aseguren su interoperabilidad, los cuales deben estar normados por la WS-I (Web Service Interoperability).

Se quiere diagnosticar cuales son los pasos a seguir para adoptar la norma WS-I (Web Service Interoperability), por lo que se requiere hacer un estudio del estado actual de los Servicios Web de Softel, específicamente los generados en el marco del proyecto APS, y cuál es su situación en relación con el estándar WS-I. Además buscar soluciones para migrar todos estos Servicios Web hacia ese estándar. Así se podría estandarizar todos estos servicios web de la empresa de Softel y eliminar los problemas que existen en la actualidad referente a la comunicación y más directamente con la creación de los wsdl de las distintas aplicaciones existentes.

Teniendo en cuenta lo antes expuesto se propone como **situación problemática:**

Los servicios web desarrollados en la empresa Softel son diversos y aunque cumplen con un estándar definido, se desea migrar los mismos a un estándar más pequeño, robusto, que mejore aspectos de la seguridad, la calidad y la interoperabilidad de los mismos.

Dado esta problemática el **problema a resolver** es el siguiente: ¿Cómo lograr que los servicios web de la empresa Softel cumplan con la norma WS-I (Web Service Interoperability)?

A partir de aquí se enmarca como **objeto de estudio**: el proceso de desarrollo de software orientado a servicios en la empresa Softel y como **campo de acción:** el proceso de elaboración de los wsdl, como elemento integrador dentro del proceso de desarrollo de software dentro de la empresa de Softel.

Teniendo en cuenta lo anteriormente planteado el **objetivo general** sería:

Realizar propuesta de procedimiento para la aplicación de las distintas definiciones/especificaciones, definidas dentro de los estándares planteados por la WS-I, durante la elaboración de las descripciones del servicio Web (wsdl).

La idea a defender es la siguiente: Con la creación del procedimiento se lograría estandarizar las descripciones de los servicios Web (wsdl), mejorando la interoperabilidad y calidad de los servicios web de la empresa Softel.

Para dar solución a los objetivos planteados en la investigación se trazaron una serie de **tareas de investigación,** las cuales se muestran a continuación:

- Realizar estudio de la situación actual de los perfiles definidos por la WS-I.
- Estudiar y valorar las herramientas que permiten comprobar la conformidad con los perfiles de la WS-I.
- Analizar los servicios web publicados en Softel, de manera concreta los desarrollados por el proyecto APS.
- Evaluar los servicios web publicados por Softel respecto a los estándares definidos por la WS-I.
- Proponer estrategia para la migración de los servicios web a la norma WS-I.

Entre los beneficios y aportes de la investigación se encuentran:

- Lograr unificar en un estándar todos los Servicios Web de Softel.
- Mejorar la interoperabilidad de los Servicios Web.
- Lograr la correcta utilización de los estándares de la WS-I.

INTRODUCCIÓN

Nuestro trabajo consta de cuatro partes fundamentales: una introducción donde se explica el diseño teórico; un capitulo uno, cuyo titulo es “Fundamentación Teórica”; un capitulo 2 nombrado “Descripciones de las Soluciones Propuestas” y un tercero llamado “Propuesta de Procedimiento”.

CAPÍTULO I
Fundamentación Teórica

En este capitulo se pretende dar una introducción al tema que será centro de atención en el transcurso de la investigación, como lograr que los wsdl queden normados según la WS-I (Web Service Interoperability). Se pretende explicar, lo más detallado posible, como es que se utilizan y la importancia que revisten para el buen funcionamiento de la comunicación, para ello el presente trabajo se centra en los objetivos trazados. Es necesario conocer los conceptos fundamentales que se deben dominar para el entendimiento de con qué se está trabajando y por qué.

1.1 Estado del arte

Softel es la empresa que ofrece soluciones informáticas para el Sistema de Salud, para lo cual dispone de profesionales con experiencia en diseño, implantación y gestión de estas soluciones. Combina la experiencia en el desarrollo e integración de soluciones informáticas para el sector de la salud, con la aplicación de modernas tecnologías. Su propósito: satisfacer las expectativas del sistema de salud, brindándole herramientas que permitan aumentar la calidad de los servicios en sus instituciones. Más de 20 años como líder en el sector de la informática en Cuba, avalan el quehacer profesional de esta institución.

La empresa Softel desea asimilar e implantar una arquitectura orientada a servicios, pero, como en cualquier proceso de implantación de nuevas tendencias tecnológicas, existen algunas dificultades: los servicios Web de la empresa son diversos, presentan problemas propios de el estándar que los define, por lo que se hace necesario hacer una migración de los wsdl de estos a un estándar específico, de forma tal que todos estén identificados por un mismo patrón y el trabajo sea mucho más eficiente. Para erradicar los problemas existentes se propone el uso de los perfiles definidos por la WS-I (Web Service Interoperability), organización vanguardia en el tema.

1.2 SOA

1.2.1 ¿Qué es SOA?

La Arquitectura Orientada a Servicios (en inglés Service-Oriented Architecture o SOA), es un concepto de arquitectura de software que define la utilización de servicios para dar soporte a los requerimientos de software del usuario.

SOA es una arquitectura de software que permite la creación y/o cambios de los procesos de negocio, a través de la composición de nuevos procesos utilizando las funcionalidades de negocio que están contenidas en la infraestructura de aplicaciones actuales o futuras (expuestas bajo la forma de Servicios Web).

1.2.2 Capas de software que define SOA

- aplicativa básica, sistemas desarrollados bajo cualquier arquitectura o tecnología, geográficamente dispersos y bajo cualquier figura de propiedad;
- de exposición de funcionalidades, donde las funcionalidades de la capa aplicativas son expuestas en forma de servicios (servicios web);
- de integración de servicios, facilitan el intercambio de datos entre elementos de la capa aplicativa orientada a procesos empresariales internos o en colaboración;
- de composición de procesos, que define el proceso en términos del negocio y sus necesidades, y que varia en función del negocio;
- de entrega, donde los servicios son desplegados a los usuarios finales.

1.2.3 Beneficios que conlleva el uso de SOA

- Mejora en los tiempos de realización de cambios en procesos.
- Facilidad para evolucionar a modelos de negocios basados en tercerización.
- Facilidad para abordar modelos de negocios basados en colaboración con otros entes (socios, proveedores).

- Poder para reemplazar elementos de la capa aplicativa SOA sin disrupción en el proceso de negocio.

1.2.4 Diseño y desarrollo de SOA

La metodología de modelado y diseño para aplicaciones SOA se conoce como análisis y diseño orientado a servicios. La arquitectura orientada a servicios es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implantación. Para que un proyecto SOA tenga éxito los desarrolladores de software deben orientarse ellos mismos, a una mentalidad de crear servicios comunes que son orquestados por clientes o middleware para implementar los procesos de negocio. El desarrollo de sistemas usando SOA requiere un compromiso con este modelo en términos de planificación, herramientas e infraestructura.

Una arquitectura orientada a servicios es más bien como un juego de servicios residentes en Internet o en una intranet, usando servicios web. Existe un juego de estándares, de los que se habla, ligados a los servicios Web:

- XML
- HTTP
- SOAP
- WSDL
- UDDI

Un sistema SOA no necesariamente necesita utilizar estos estándares para ser "orientado a servicios" pero es altamente recomendable su uso.

1.3 WSDL

1.3.1 ¿Qué es WSDL?

WSDL son las siglas de Web Services Description Language, un formato XML que se utiliza para describir servicios Web.

WSDL describe la interfaz pública a los servicios Web. Está basado en XML y describe la forma de comunicación, es decir, los requisitos del protocolo y los formatos de los mensajes necesarios para interactuar con los servicios listados en su catálogo. Las operaciones y mensajes que soporta se describen en abstracto y se ligan después al protocolo concreto de red y al formato del mensaje.

WSDL se usa a menudo en combinación con SOAP y XML Schema. Un programa cliente que se conecta a un servicio web puede leer el WSDL para determinar que funciones están disponibles en el servidor. Los tipos de datos especiales se incluyen en el archivo WSDL en forma de XML Schema. El cliente puede usar SOAP para hacer la llamada a una de las funciones listadas en el WSDL. [1]

1.3.2 WSDL para la documentación de Servicios Web

El esquema XML por sí solo no puede describir totalmente un Servicio Web.

El lenguaje de descripción de servicios Web (WSDL, Web Service Description Language) es un dialecto basado en XML sobre el esquema que describe un servicio Web. Un documento WSDL proporciona la información necesaria al cliente para interaccionar con el servicio Web. WSDL es extensible y se pude utilizar para describir, prácticamente, cualquier servicio de red.

Dado que los protocolos de comunicaciones y los formatos de mensajes están estandarizados en la comunidad del Web, cada día aumenta la posibilidad e importancia de describir las comunicaciones de forma estructurada. WSDL afronta esta necesidad definiendo una gramática XML que describe los servicios de red como colecciones de puntos finales de comunicación capaces de intercambiar mensajes. Las definiciones de servicio de WSDL proporcionan documentación para sistemas distribuidos y sirven como fórmula para automatizar los detalles que toman parte en la comunicación entre aplicaciones.

Los documentos WSDL definen los servicios como colecciones de puntos finales de red o puertos. En WSDL, la definición abstracta de puntos finales y de mensajes se separa de la instalación concreta de red o de los enlaces del formato de datos. Esto permite la reutilización de definiciones abstractas: mensajes, que son descripciones abstractas de los datos que se están intercambiando y tipos de puertos, que son colecciones abstractas de operaciones. Las especificaciones concretas del protocolo y del formato de datos para un tipo de puerto determinado constituyen un enlace reutilizable. Un puerto se define por la asociación de una dirección de red y un enlace reutilizable; una colección de puertos define un servicio.

Por esta razón, un documento WSDL utiliza los siguientes elementos en la definición de servicios de red: [2]

- **Types**: contenedor de definiciones del tipo de datos que utiliza algún sistema de tipos (por ejemplo XSD).
- **Message**: definición abstracta y escrita de los datos que se están comunicando.
- **Operation**: descripción abstracta de una acción admitida por el servicio.
- **Port Type**: conjunto abstracto de operaciones admitidas por uno o más puntos finales.
- **Binding**: especificación del protocolo y del formato de datos para un tipo de puerto determinado.
- **Port**: punto final único que se define como la combinación de un enlace y una dirección de red.
- **Service**: colección de puntos finales relacionados.

A continuación se detallan algunos de estos elementos:

Elemento types:

Esta etiqueta define las estructuras de datos que se utilizarán para construir los mensajes tanto de petición como de respuesta. Estas estructuras de datos pueden construirse con cualquier lenguaje, pero lo más normal es hacerlo con XML Schema. Este apartado es el más complicado sobre todo cuando se tengan que construir estructuras de datos muy complejas. Si se usa esquema de XML para definir los tipos que contiene el elemento Types el elemento schema aparecerá inmediatamente como elemento hijo.

Elemento message:

Describe los mensajes que se van a intercambiar entre el cliente y el Servicio Web. Un mensaje puede estar dividido en varias partes, por ejemplo, si en un mensaje se quieren enviar datos y una imagen. El elemento Message proporciona una abstracción común para el paso de mensajes entre el cliente y el servidor. Como puede utilizar múltiples formatos de de definición de esquema en documento WSDL es necesario disponer de un mecanismo común de identificar los mensajes. El elemento Message proporciona este nivel común de abstracción al que se hará referencia en otras partes del documento WSDL.

Elemento portType:

Define el conjunto de operaciones que soporta el Servicio Web. Una operación no es más que un grupo de mensajes que serán intercambiados. Cada operación puede enviar o recibir al menos un mensaje cada vez.

El elemento porType contiene un conjunto de operaciones abstractas que representan los tipos de correspondencia que pueden producirse entre el cliente y el servidor.

Un tipo puerto se compone de un conjunto de elementos operation que define una determinada acción. Los elementos operation se componen de mensajes definidos en el documento WSDL. WSDL define cuatro tipos de operaciones denominadas tipo operaciones: [3]

- Request-response (petición-respuesta) el cliente realiza una petición y el servidor envía la correspondiente respuesta.
- One-way (un-sentido) el cliente envía un mensaje pero no recibe una respuesta del servidor indicando el resultado del mensaje procesado.
- Solicit-response (solicitud-respuesta) La contraria a la operación petición-respuesta. El servidor envía una petición y el cliente le envía de vuelta una respuesta.
- Notification (Notificación) La contraria a la operación un-sentido, el servidor envía una petición, sin recibir respuesta por parte del cliente.

Elemento binding:

El elemento binding contiene las definiciones de la asociación de un protocolo como SOAP a un determinado bindingType. Las definiciones binding especifican detalles de formatos del mensaje y el protocolo. Por ejemplo, la información de asociación especifica si se puede acceder a una instancia de un portType de forma RPC.

En el caso de la unión SOAP, esta soporta dos estilos de comunicación: RPC y Document. El estilo RPC soporta formación y deformación automática de mensajes, permitiendo a los desarrolladores expresar una solicitud como una llamada a método con un conjunto de parámetros, que devuelve una respuesta que contiene un valor de retorno. El estilo Document no soporta formación automática de mensajes. Asume

que los contenidos del mensaje SOAP son datos XML bien-formateados. Este tipo de unión también requiere que se especifique como se expresan los mensajes en XML. Se usan valores literales o tipos de datos encoded. El atributo use='literal' indica que el entorno de ejecución SOAP debería enviar el XML como se le ha proporcionado. El atributo use='encoded' indica que el entorno de ejecución SOAP debería serializar los datos usando un estilo de codificación particular.

Elemento service:

Un servicio es un grupo de puertos relacionados, los cuales se definen en el elemento service. Un puerto es un extremo concreto de un Servicio Web al que se hace referencia por una dirección única. Los puertos que se definen en determinado servicio son independientes.

1.4 Servicios Web

Existen múltiples definiciones sobre lo que son los Servicios Web, lo que muestra su complejidad a la hora de dar una adecuada definición que englobe todo lo que son e implican. Una posible, sería hablar de ellos como un conjunto de aplicaciones o de tecnologías con capacidad para interoperar en la Web o una colección de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. Estas aplicaciones o tecnologías intercambian datos entre sí con el objetivo de ofrecer unos servicios.

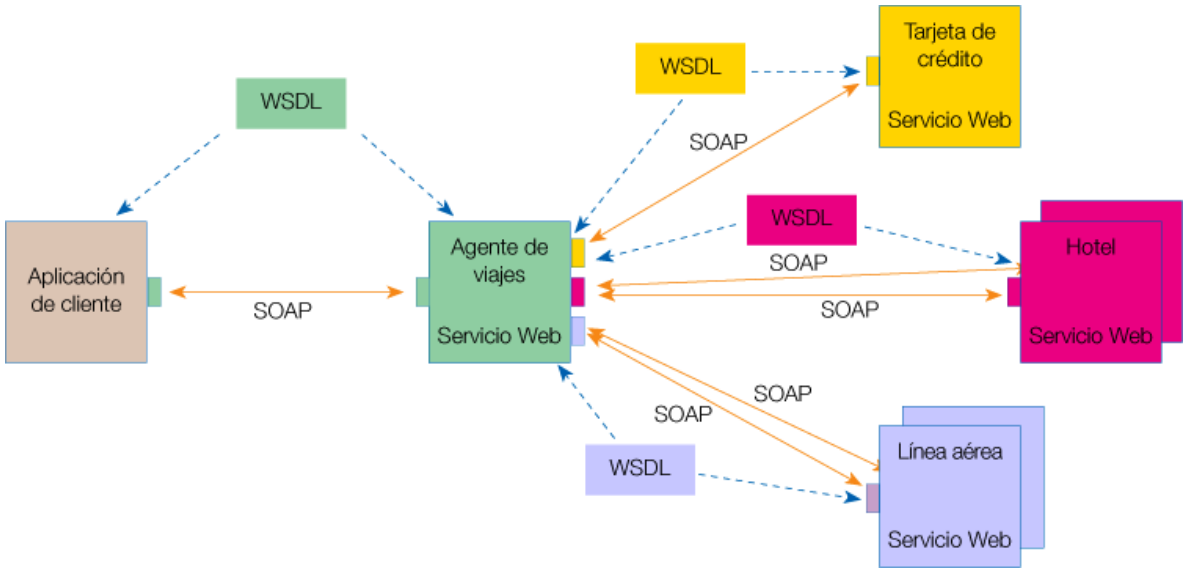
Distintas aplicaciones de software desarrolladas en diferentes lenguajes de programación y ejecutadas sobre cualquier plataforma, pueden utilizar los servicios Web para intercambiar datos en redes de ordenadores como Internet. La interoperabilidad se consigue mediante la adopción de estándares abiertos. Los proveedores ofrecen sus servicios como procedimientos remotos y los usuarios solicitan un servicio llamando a estos procedimientos a través de la Web.

Las organizaciones OASIS y W3C son los comités responsables de la arquitectura y reglamentación de los servicios Web. Para mejorar la interoperabilidad entre distintas implementaciones de servicios Web se ha creado el organismo WS-I, encargado de desarrollar diversos perfiles para definir de manera más exhaustiva estos estándares.

1.4.1 ¿Cómo funcionan?

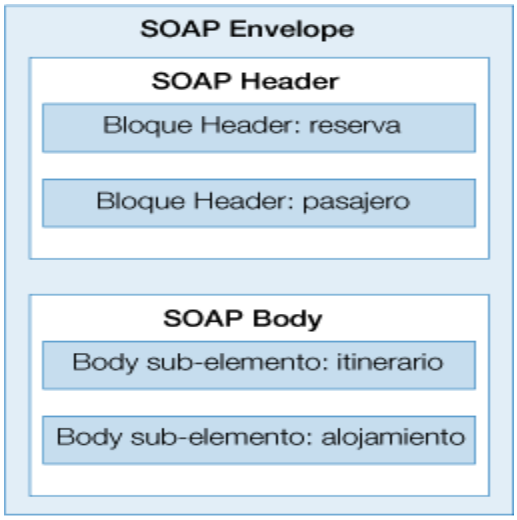
Para comprender el funcionamiento de los servicios web, se muestra un ejemplo:

Según el ejemplo del gráfico mostrado a continuación, un usuario (que juega el papel de cliente dentro de los Servicios Web), a través de una aplicación, solicita información sobre un viaje que desea realizar haciendo una petición a una agencia de viajes que ofrece sus **servicios** a través de Internet. La agencia de viajes ofrecerá a su cliente (usuario) la información requerida. Para proporcionar al cliente la información que necesita, esta agencia de viajes solicita a su vez información a otros recursos (otros Servicios Web) en relación con el hotel y la compañía aérea. La agencia de viajes obtendrá información de estos recursos, lo que la convierte a su vez en cliente de esos otros Servicios Web que le van a proporcionar la información solicitada sobre el hotel y la línea aérea. Por último, el usuario realizará el pago del viaje a través de la agencia de viajes que servirá de intermediario entre el usuario y el servicio Web que gestionará el pago.



En este proceso están presentes diferentes tecnologías:

Se tendría a SOAP (Protocolo Simple de Acceso a Objetos), para especificar el formato de los mensajes, donde los mismos estan compuestos por un envelope (sobre), cuya estructura esta formada por los elementos header (cabecera) y body (cuerpo). [4]



Estructura de los mensajes

Por otro lado, WSDL (Lenguaje de Descripción de Servicios Web), al cual ya se ha hecho referencia anteriormente, en el presente capitulo.

1.4.2 ¿Para qué sirven?

Estos servicios proporcionan mecanismos de comunicación estándares entre diferentes aplicaciones, que interactúan entre sí para presentar información dinámica al usuario. Para proporcionar interoperabilidad y extensibilidad entre estas aplicaciones, y que al mismo tiempo sea posible su combinación para realizar operaciones complejas, es necesaria una arquitectura de referencia estándar, entiéndase lo mismo como el uso de diferentes estándares que aseguren la funcionalidad del servicio Web.

1.4.3 Estándares empleados

- ***Web Services Protocol Stack*:** Así se denomina al conjunto de servicios y protocolos de los servicios Web.
- ***XML* (Extensible Markup Language):** Es el formato estándar para los datos que se vayan a intercambiar. Es un subconjunto simplificado del SGML el cual fue diseñado principalmente para documentos Web. Deja a los diseñadores crear sus propias “etiquetas” o "tags" (Ej: <libro>), habilitando la definición, transmisión, validación, y la interpretación de datos entre aplicaciones y entre organizaciones. El HTML y el XML tienen funciones diferentes, el HTML tiene por objeto

mostrar información, mientras que el XML se ocupa de la información propiamente dicha (el contenido).

Algunas ventajas: [5]

- Es una arquitectura abierta y extensible. No se necesita versiones para que pueda funcionar en futuros navegadores.
- Integración de los datos de las fuentes más dispares. Se podrá hacer el intercambio de documentos entre las aplicaciones tanto en el propio PC como en una red local.
- Datos compuestos de múltiples aplicaciones. La extensibilidad y flexibilidad de este lenguaje permitirá agrupar una variedad amplia de aplicaciones, desde páginas web hasta bases de datos.
- Los motores de búsqueda devolverán respuestas más adecuadas y precisas, ya que la codificación del contenido web en XML consigue que la estructura de la información resulte más accesible.

❖ ***SOAP* (Simple Object Access Protocol) o *XML-RPC* (XML Remote Producer Call):** Protocolos sobre los que se establece el intercambio. En el caso de SOAP decir que es un protocolo de comunicación, el cual permite la comunicación entre aplicaciones a través de mensajes por medio de Internet. Es independiente de la plataforma y del lenguaje. Esta basado en XML y es la base principal de los servicios Web. Los mensajes SOAP son un documento XML propiamente dicho.

Historia de SOAP:

Este protocolo deriva de un protocolo creado por David Winer, XML-RPC en 1998. Con el mismo se pedían realizar RPC o remote procedure calls, es decir, tanto en cliente o servidor, se pueden realizar peticiones mediante http a un servidor web. Los mensajes debían tener un formato determinado empleando XML para encapsular los parámetros de la petición. Con el paso del tiempo el proyecto iniciado por David Winer interesó a importantes multinacionales entre las que se encuentran IBM y Microsoft y de este interés por XML-RPC, se desarrolló SOAP.

Ventajas de SOAP:

- **No esta asociado con ningún lenguaje**: los desarrolladores involucrados en nuevos proyectos pueden elegir desarrollar con el último y mejor lenguaje de programación que exista pero los desarrolladores responsables de mantener antiguas aflicciones heredadas podrían no poder hacer esta elección sobre el lenguaje de programación que utilizan. SOAP no especifica una API, por lo que la implementación de la API se deja al lenguaje de programación, como en Java, y la plataforma como Microsoft .Net.

- **No se encuentra fuertemente asociado a ningún protocolo de transporte**: la especificación de SOAP no describe como se deberían asociar los mensajes de SOAP con HTTP. Un mensaje de SOAP no es más que un documento XML, por lo que puede transportarse utilizando cualquier protocolo capaz de transmitir texto.

- **No está atado a ninguna infraestructura de objeto distribuido:** la mayoría de los sistemas de objetos distribuidos se pueden extender, y ya lo están alguno de ellos para que admitan SOAP.

- **Aprovecha los estándares existentes en la industria**: los principales contribuyentes a la especificación SOAP evitaron, intencionadamente, reinventar las cosas. Optaron por extender los estándares existentes para que coincidieran con sus necesidades. Por ejemplo, SOAP aprovecha XML para la codificación de los mensajes, en lugar de utilizar su propio sistema de tipo que ya están definidas en la especificación esquema de XML. Y como ya se ha mencionado SOAP no define un medio de trasporte de los mensajes; los mensajes de SOAP se pueden asociar a los protocolos de transporte existentes como HTTP y SMTP.

- **Permite la interoperabilidad entre múltiples entornos**: SOAP se desarrolló sobre los estándares existentes de la industria, por lo que las aplicaciones que se ejecuten en plataformas con dicho estándares pueden comunicarse mediante mensaje SOAP con aplicaciones que se ejecuten en otras plataformas. Por ejemplo, una aplicación de escritorio que se ejecute en una PC puede comunicarse con una aplicación del back-end ejecutándose en un mainframe capaz de enviar y recibir XML sobre HTTP.

- Otros protocolos: los datos en XML también pueden enviarse de una aplicación a otra mediante protocolos normales como ***HTTP*** (Hypertext Transfer Protocol), ***FTP*** (File Transfer Protocol), o ***SMTP*** (Simple Mail Transfer Protocol).

- ***WSDL*** **(Web Services Description Languages):** Es el lenguaje de la interfaz pública para los servicios Web. Es una descripción basada en XML de los requisitos funcionales necesarios para establecer una comunicación con los servicios Web, indica cuales son las interfaces que provee el servicio Web y los tipos de datos necesarios para la utilización del mismo.

- ***UDDI*** **(Universal Description, Discovery and Integration):** Protocolo para publicar la información de los servicios Web. Permite comprobar qué servicios web están disponibles. Es un modelo de directorios para servicios Web. Es una especificación para mantener directorios estandarizados de información acerca de los servicios Web, sus capacidades, ubicación, y requerimientos en un formato reconocido universalmente. UDDI utiliza WSDL para describir las interfaces de los servicios Web. Es un lugar en el cual se buscan cuales son los servicios Web disponibles, una especie de directorio en el que se encuentran los servicios Web publicados y publicar los Web Services que se desarrollen.

- ***WS-Security*** **(Web Service Security):** Protocolo de seguridad aceptado como estándar por OASIS (Organization for the Advancement of Structured Information Standards). Garantiza la autenticación de los actores y la confidencialidad de los mensajes enviados. [6]

1.4.4 Razones para crear servicios Web

La principal razón para usar servicios Web es que se basan en HTTP sobre TCP (Transmission Control Protocol) en el puerto 80. Dado que las organizaciones protegen sus redes mediante cortafuegos (*Firewalls)*, que filtran y bloquean gran parte del tráfico de Internet, cierran casi todos los puertos TCP, salvo el 80 que es precisamente el que usan los navegadores. Los servicios Web se enrutan por este puerto, por la simple razón de que no resultan bloqueados.

Otra razón es que, antes de que existiera SOAP, no había buenas interfaces para acceder a las funcionalidades de otros ordenadores en red. Las que había eran específicamente con ese fin y poco

conocidas, tales como EDI (Electronic Data Interchange), RPC, u otras APIs (Application Programming Interface).

Una tercera razón por la que los servicios Web son muy prácticos es que pueden aportar gran independencia entre la aplicación que usa el servicio Web y el propio servicio. De esta forma, los cambios a lo largo del tiempo en uno no deben afectar al otro. Esta flexibilidad será cada vez más importante, dado que la tendencia a construir grandes aplicaciones a partir de componentes distribuidos más pequeños es cada día más acusada.

1.4.5 Requisitos de un Servicio Web

Interoperabilidad: Un servicio remoto debe permitir su utilización por clientes de otras plataformas.

Amigabilidad con Internet: La solución debe poder funcionar para soportar clientes que accedan a los servicios remotos desde internet.

Interfaces fuertemente tipadas: No debería haber ambigüedad acerca del tipo de dato enviado y recibido desde un servicio remoto. Más aún, los tipos de datos definidos en el servicio remoto deben poderse corresponder razonablemente bien con los tipos de datos de la mayoría de los lenguajes de programación procedimentales.

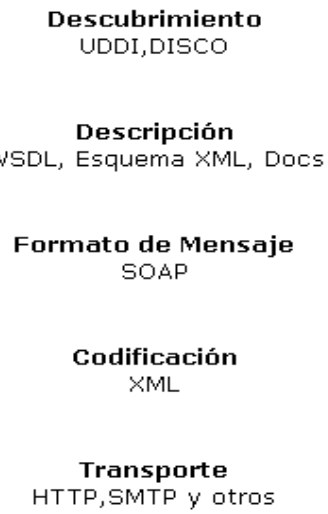
Posibilidad de aprovechar los estándares de Internet existentes: La implementación del servicio remoto debería aprovechar estándares de Internet existentes tanto como sea posible y evitar reinventar soluciones a problema que ya se han resuelto. Una solución construida sobre un estándar de Internet ampliamente adoptado puede aprovechar un conjunto de herramientas y productos existentes creados para dicha tecnología.

Soporte para cualquier lenguaje: La solución no debería ligarse a un lenguaje de programación particular, Java RMI, por ejemplo, esta ligada completamente a lenguaje Java. Sería muy difícil invocar la funcionalidad de un objeto Java remoto desde Visual Basic o PERL. Un cliente debería ser capaz de implementar un nuevo servicio Web existente independientemente del lenguaje de programación en el que se halla escrito el cliente. [7]

Soporte para cualquier infraestructura de componente distribuida: La solución no debe estar fuertemente ligada a una infraestructura de componentes en particular.

1.4.6 Bloques Constructivos de Servicios Web

En la siguiente figura se muestran los bloques constructivos principales necesarios para facilitar las comunicaciones remotas entre aflicciones.



Descubrimiento: La aplicación cliente que necesita acceder a la funcionalidad que expone un Servicio Web necesita una forma de resolver la ubicación de servicio remoto. Se logra mediante un proceso llamado, normalmente descubrimiento (discovery). El descubrimiento se puede proporcionar mediante un directorio centralizado así como por otros métodos ad hoc.

Descripción: Una vez que se ha resuelto el extremo de un servicio Web dado, el cliente necesita suficiente información para interactuar adecuadamente con el mismo. La descripción de un servicio Web implica metadatos estructurados sobre la interfaz que intenta utilizar la aplicación cliente así como documentación escrita sobré el servicio Web incluyendo ejemplo de uso.

Formato del mensaje: Para el intercambio de datos, el cliente y el servidor tienen que estar de acuerdo en un mecanismo común de codificación y formato de mensaje. El uso de un mecanismo estándar de codificar los datos asegura que los datos que codifica el cliente los interpretará correctamente el servidor.

Codificación: Los datos que se trasmiten entre el cliente y el servidor necesitan codificarse en un cuerpo de mensaje.

Transporte: Una vez se ha dado formato al mensaje y se han serializado los datos en el cuerpo del mensaje se debe transferir entre el cliente y el servidor utilizando algún protocolo de transporte.

1.4.7 Ventajas de los servicios Web

- Aportan interoperabilidad entre aplicaciones de software independientemente de sus propiedades o de las plataformas sobre las que se instalen.
- Los servicios Web fomentan los estándares y protocolos basados en texto, que hacen más fácil acceder a su contenido y entender su funcionamiento.
- Al apoyarse en HTTP, los servicios Web pueden aprovecharse de los sistemas de seguridad firewall sin necesidad de cambiar las reglas de filtrado.
- Permiten que servicios y software de diferentes compañías ubicadas en diferentes lugares geográficos puedan ser combinados fácilmente para proveer servicios integrados. [8]

Conclusiones

Se lograron esclarecer los conceptos fundamentales que se tratarán a lo largo de la investigación, tratándose de manera especial el de WSDL (Web Service Description Lenguage) el cual es el tema fundamental al que está enfocado el trabajo. Además se dio una pequeña introducción de aspectos fundamentales como servicios web, arquitectura SOA, que son términos básicos en la investigación y que son ampliamente utilizados a lo largo de la misma.

CAPÍTULO II
Descripción de las soluciones que han sido propuestas

El presente capitulo hará énfasis, al punto principal de nuestra investigación, que no es más que el enfoque a la WS-I, su concepto, surgimiento, perfiles que define entre otros aspectos. Dentro de la arquitectura SOA para la implementación de Servicios Web, la interoperabilidad es tal vez el principio más importante. Como método de implementación de SOA, Web Services debe ofrecer importantes beneficios de interoperabilidad, y permitir la ejecución de servicios Web distribuidos en múltiples plataformas de software y arquitecturas de hardware, de ahí la necesidad del estudio de la WS-I, que es la encargada en ese sentido.

2. WS-I

2.1 ¿Qué es?

La **Organización para la Interoperabilidad de Servicios Web** (*Web Services Interoperability Organization*) tiene como objetivo fomentar y promover la Interoperabilidad de Servicios Web (*Web Services Interoperability* - **WS-I**) sobre cualquier plataforma, sobre aplicaciones, y sobre lenguajes de programación.

Su intención es ser un integrador de estándares para ayudar al avance de los servicios web de una manera estructurada y coherente.

2.2 Estructura

La WS-I organiza Grupos de Trabajo (Working Groups, WG), que se centran en las especificaciones y documentación de servicios Web concretos. Cada Grupo de Trabajo tiene encomendada la generación de un documento específico que ayudará a los desarrolladores en su búsqueda de interoperabilidad. Entre los documentos producidos por la WS-I se incluyen perfiles, escenarios de uso y casos de uso, aplicaciones de ejemplo y herramientas de test. [9]

2.3 Estándares de Perfiles que define

Existen varios estándares que necesitan ser coordinados, para obtener un buen resultado en lo referente a la interoperabilidad de los servicios. Dichos estándares se desarrollan en paralelo y de manera independiente por la WS-I, bajo el concepto '**perfil'** (*profile*)

La WS-I define un perfil como: *Un conjunto de definiciones/especificaciones comúnmente aceptadas por la industria y a partir del apoyo a estándares basados en XML, junto con un conjunto de indicaciones que recomiendan cómo se deben usar las especificaciones para desarrollar servicios web interoperables entre sí*.

Entre los perfiles que define se encuentran:

- WS-I Basic Profile.
- WS-I Basic Security Profile.

Estos perfiles ofrecen la guía de diseño necesaria para evitar los problemas de interoperabilidad más comunes, que están presentes en los servicios Web. Para cada perfil de WS-I se dispone de un catálogo de ejemplos y herramientas de test. Los ejemplos enseñan a implementar una aplicación plenamente compatible con las directrices WS-I y las herramientas de test, permiten comprobar el grado de conformidad de nuestras aplicaciones. [10]

2.4 Perfiles de servicios Web

La WS-I ya ha terminado su primer perfil, llamado Basic Profile (BP) version 1.0 (en inglés), en Abril del 2004. El Perfil Básico ("PB") establece que los servicios Web propiamente dichos utilizan SOAP, Web Services Description Language (WSDL), y UDDI (Universal Description, Discovery, and Integration) y el mismo aporta guías de alto valor sobre cómo utilizar correctamente cada uno de ellos.

Junto con la guía del Perfil Básico, la WS-I también ha publicado los Escenarios de Uso y una aplicación de ejemplo completa, basada en un escenario de cadena de aprovisionamiento (completada con casos de uso, diseño y archivos de implementación).La WS-I ha publicado también una serie de herramientas de prueba para verificar la conformidad con el Perfil Básico en las aplicaciones. Ya existen resultados con el

trabajo realizado en el Perfil Básico, contribuyendo a mejorar la interoperabilidad entre las implementaciones.

La WS-I trabaja también en perfiles adicionales, entre ellos están:

EL Borrador del Perfil Básico de Seguridad 1.0 (en inglés), orientado como una guía sobre seguridad en servicios Web. El Perfil Básico de Seguridad cubre aspectos como la seguridad en servicios Web (WS-Security),

El *Approval Draft for the Attachments Profile 1.0* (en inglés), que es una guía sobre el uso de mensajes SOAP con archivos adjuntos, XML 1.0 y espacios de nombres, y MIME para adjuntar recursos no XML a los mensajes SOAP.

La guía propuesta por el Perfil Básico se organiza en cuatro categorías: mensajería, descripción del servicio, publicación y descubrimiento del servicio, y seguridad de servicios Web. No obstante, la parte más importante de las guías se centra en el uso correcto de definiciones de diseño WSDL para la descripción del servicio Web, puesto que las incompatibilidades a este nivel pueden, provocar problemas muy importantes. [11]

2.5 Categorías

2.5.1 Mensajería

La guía sobre intercambio de mensajes SOAP aclara algunas de las ambigüedades que existen en las especificaciones SOAP y HTTP. Por ejemplo, la guía establece que los receptores deben poder procesar mensajes SOAP que contienen la marca BOM (Byte order mark) en formato Unicode, y que todos los mensajes deben utilizar la codificación de caracteres UTF-8 ó UTF-16. Establece que los mensajes no pueden contener instrucciones de procesamiento o DTDs (Document Type Definition). Además especifica cómo se estructura correctamente el elemento de fallo SOAP y cómo se utilizan los mensajes soap: mustUnderstand y atributos soap: encodingStyle en los mensajes SOAP.

Ahora, se muestran algunas piezas de la guía de buenas prácticas que tienen una utilidad directa para todos los desarrolladores de servicios Web.

La primera está relacionada con el procesamiento correcto de los bloques de cabecera obligatorios. De acuerdo con el perfil "este requisito garantiza que no se producirán efectos colaterales no deseados como resultado de la aparición de un bloque obligatorio de cabecera después de haber procesado otras partes del mensaje. El Perfil exige que los receptores generen un Error cuando encuentren bloques de cabecera que consideran que no van dirigidos a ellos. Cuando se genera un Error, no se deben continuar el procesamiento."

El uso de cookies para implementar servicios Web con control de estado es otro de los temas que no se definen explícitamente en las especificaciones de mensajería de SOAP. El mecanismo de gestión de estado HTTP ("Cookies") permite la creación de sesiones que mantienen el estado entre clientes y servidores Web. El PB no rechaza el uso de cookies, pero limita su utilización. La guía establece fundamentalmente que no se exigirá el uso de cookies para realizar una operación. En lugar de ello, si se utilizan, deben servir como punto de referencia que se empleará para optimizar el procesamiento sin afectar directamente a la ejecución del servicio Web. Y los consumidores no tendrán que saber sobre valores de las cookies, solo necesitan pasarlas en peticiones futuras.

2.5.2 Descripción del servicio Web

El Perfil Básico establece que se utilizará WSDL para describir los servicios Web. WSDL se diseño ex-profeso como entorno abierto y flexible con muchas opciones a disposición de los programadores. Como resultado de esta filosofía general, WSDL se ha convertido en el punto de fricción de la interoperabilidad de los servicios Web. El PB de la WS-I ha resuelto este problema eliminando muchas de las opciones y reduciendo el número de detalles que se deben acordar en la especificación.

Algunas de las clarificaciones realizadas por el PB sobre la especificación de WSDL, son mencionadas a continuación, por ejemplo: se resuelven muchas de las ambigüedades existentes alrededor del uso de WSDL y los elementos "import" de los esquemas XML y los atributos "targetNamespace", se aclara también qué se hace con temas, como el orden de los parámetros, operaciones unidireccionales, elementos del empaquetamiento del mensaje de respuesta, descriptores de error y validación de los mensajes.

El Perfil Básico impone restricciones sobre la especificación WSDL, por ejemplo: deshabilita el uso de sistemas ajenos a los Esquemas XML. Además rechaza el uso de operaciones petición-respuesta y

notificación, la sobrecarga del nombre de operación, los enlaces MIME y HTTP GET/POST, el uso de transportes distintos de HTTP, las codificaciones incluyendo la codificación SOAP), la mezcla de estilos dentro de un enlace único y el requerimiento de extensiones WSDL a medida.

En general, un modelo de desarrollo basado desde el principio en WSDL puede aumentar su interoperabilidad en el momento en que permita el control total sobre el diseño WSDL. El empezar con WSDL permite validar si se siguen todas las guías sobre WSDL del PB antes de generar el código de implementación.

2.5.3 Publicación, descubrimiento y seguridad del servicio Web

Cuando hay que publicar y descubrir servicios Web, el Perfil Básico destaca el uso de UDDI para este fin, aunque no es necesario el uso del mismo, para lograr la compatibilidad con el PB.

La guía además ofrece directrices para el uso de HTTPS como forma de dotar de seguridad a los servicios Web actualmente. Básicamente establece que los servicios Web pueden requerir HTTPS en un punto final en concreto. Con HTTPS, los puntos finales del servicio Web pueden garantizar la autenticación en el servidor, la integridad y la privacidad para todo el proceso de intercambio de mensajes. El perfil también establece que los servicios Web pueden exigir la autenticación de cliente (autenticación mutua). Y puesto que HTTPS es un estándar maduro y de amplia difusión, su uso permite disponer de funciones básicas de seguridad para los servicios Web, lo que puede ser un punto muy importante a favor de la interoperabilidad. [12]

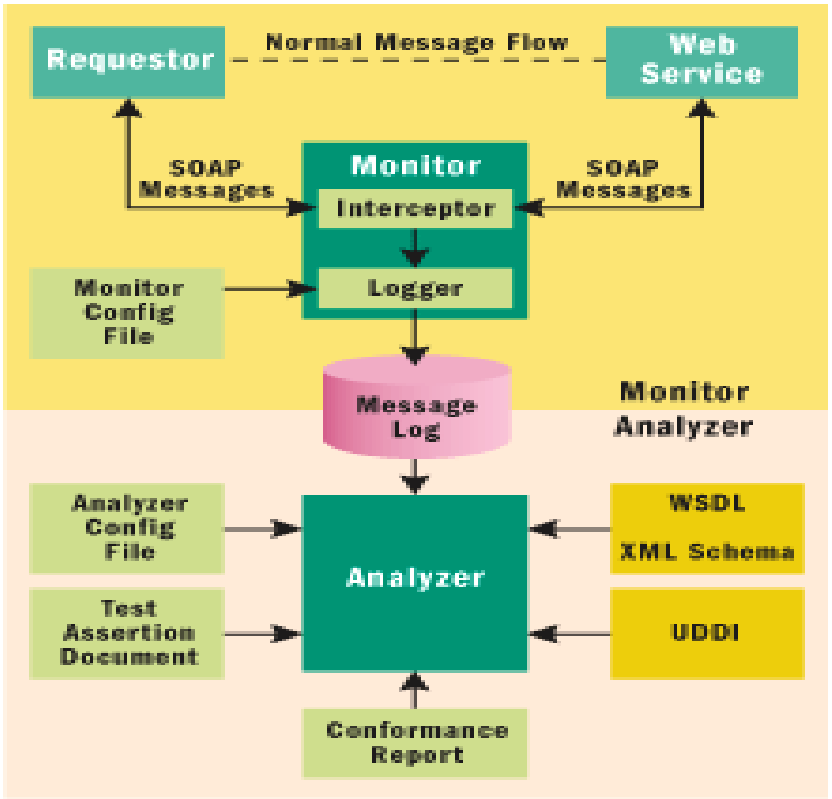
2.6 Conformidad con el Perfil Básico

Cuando un servicio Web es conforme con el PB, las posibilidades de interoperabilidad son mayores. Pero ¿cómo se sabe si un servicio Web es conforme? La conformidad es algo que realmente tiene que evaluarlo el desarrollador de servicios Web. Las herramientas de prueba pueden ayudar a validar los diseños WSDL y probar los mensajes SOAP en tiempo de ejecución. Se debe tener presente que las herramientas no pueden validar cada regla establecida por el PB, puesto que varias de ellas se refieren a especificaciones externas y comportamientos en tiempo de ejecución. Las herramientas de test tienen que entenderse como un buen indicador de conformidad y no como una herramienta de certificación oficial.

2.7 Herramientas de test de conformidad

Las herramientas de Test de la WS-I son un valioso recurso para los desarrolladores que diseñan e implementan servicios Web. Se pueden descargar estas herramientas desde el sitio web de la WS-I, disponibles en versiones C# y Java. El uso de estas herramientas durante el diseño y el desarrollo contribuye a conseguir la conformidad.

La arquitectura de las principales herramientas disponibles, el monitor y el analizador, se muestra en la figura que se muestra a continuación**.** El monitor utiliza una técnica de derivación de tráfico en un puerto sencilla, para interceptar y registrar los mensajes SOAP que se intercambian entre el emisor y el receptor. El analizador se utiliza para analizar el archivo de log junto con la correspondiente definición WSDL y las entradas UDDI. El analizador genera un informe describiendo el nivel de conformidad con el PB.



Si se desea información completa sobre el uso de estas herramientas en .NET, se sugiere la lectura del documento *"Using the WS-I Test Tools"* .También puede revisar la "Guía del Usuario de las Testing Tools" para informarse sobre su instalación y configuración.

2.8 Aplicaciones de ejemplo

Algunos de los documentos que acompañan al Perfil Básico forman parte de una aplicación de ejemplo completa, que muestra un diseño e implementación conformes. La WS-I ha publicado un documento llamado *"Supply Chain Management Use Cases"* (formato pdf, en inglés), que muestra una definición de alto nivel de una aplicación SCM (Supply Chain Management). Además ha publicado otro documento, titulado *"Sample Application Supply Chain Management Architecture"* (formato pdf, en inglés), que detalla el diseño técnico e implementación de la aplicación de ejemplo. La arquitectura incluye un Esquema XML normalizado y definiciones WSDL que muestran un diseño de contrato adecuado. La aplicación de ejemplo se ha modelado después de los casos de uso de SCM y su objetivo es demostrar cómo diseñar, implementar y poner en marcha servicios Web conformes con el PB.

2.9 Definiciones y especificaciones planteadas por el Perfil Básico

En este aspecto es donde se prestará mayor atención, pues constituye el punto principal para lograr la tan deseada interoperabilidad de los servicios Web. Se explicarán cada una de la pautas a seguir para la realización de las descripciones del servicio, apoyándose en las indicaciones del Perfil Básico. A continuación se relacionan una serie de definiciones y especificaciones que se deben seguir para el buen desarrollo de las descripciones de los servicios Web: [13]

2.9.1- Verificación de los distintos namespace en el documento

En este punto se debe chequear que los namespace definidos en la descripción estén acordes con los que se relacionan a continuación:

Prefijo	Namespace URI	Definición
wsdl	http://schemas.xmlsoap.org/wsdl/	WSDL namespace para el WSDL framework.
soap	http://schemas.xmlsoap.org/wsdl/soap/	WSDL namespace para el WSDL SOAP binding.
http	http://schemas.xmlsoap.org/wsdl/http/	WSDL namespace para el WSDL HTTP GET & POST binding.

mime	http://schemas.xmlsoap.org/wsdl/mime/	WSDL namespace para el WSDL MIME binding.
soapenc	http://schemas.xmlsoap.org/soap/encoding/	Encoding namespace como está definido por SOAP 1.1
soapenv	http://schemas.xmlsoap.org/soap/envelope/	Envelope namespace como está definido por SOAP 1.1
xsi	http://www.w3.org/2001/XMLSchema-instance	Instance namespace como está definido por XSD
xsd	http://www.w3.org/2001/XMLSchema	Schema namespace como esta definido por XSD
tns	(varios)	Este namespace prefijado (tns) es usado como una convención para referirse al documento en cuestión.

2.9.2- Descripción requerida

a) Ya sea la descripción WSDL 1.1 de una *instancia*, su *UDDI*, o ambos, *deben estar disponibles* para un consumidor autorizado cuando se solicite.

Esto quiere decir que si un consumidor autorizado pide una descripción del servicio, tanto la descripción, como su UDDI deben estar disponibles para ese consumidor.

2.9.3- Estructura del Documento

WSDL 1.1 define una estructura basada en XML para la descripción de los servicios de Web. El Perfil promulga el uso de esa estructura, y coloca las siguientes restricciones en su uso:

2.9.3.1- WSDL Definiciones de los Esquemas

a) Una *descripción* utilizando al WSDL namespace ("wsdl" colocado como prefijo en este Perfil) *debe ser válida* según el Esquema XML encontrado en "http://ws-i.org/profiles/basic/1.1/wsdl-2004-08-24.xsd".

b) Una *descripción* utilizando al WSDL SOAP Binding namespace ("wsoap12" colocado como prefijo en este Perfil) *debe ser válida* según el Esquema XML encontrado en

"http://ws-i.org/profiles/basic/1.1/wsdlsoap-2004-08-24.xsd".

2.9.3.2- WSDL y los esquemas importados

a) Una *descripción* sólo *debe usar* la declaración del WSDL "import" para importar otra descripción WSDL.

b) En una *descripción*, el atributo “namespace” del wsdl: import *no debe ser* un URI relativo.

c) Para importar definiciones de un Esquema XML, una *descripción debe usar* la declaración "import" del Esquema XML.

d) Una *descripción debe usar* la declaración "import" del Esquema XML sólo dentro del elemento xsd: schema de la sección de tipos.

e) En una *descripción* el atributo “schemaLocation” de un elemento xsd: import *no debe separarse en sus partes* para cualquier documento cuya raíz el elemento no es "esquema" del namespace "http://www.w3.org/2001/XMLSchema".

f) Un Esquema XML directamente o indirectamente importado por una *descripción puede incluir* el Unicode Byte Order Mark (BOM).

g) Un Esquema XML directamente o indirectamente importado por una *descripción debe usar* ya sea codificación UTF-8 o UTF-16.

h) Un Esquema XML directamente o indirectamente importado por una *descripción debe utilizar* la versión XML 1.0, recomendada por la W3C.

2.9.3.3- WSDL y la estructura del atributo location a importar

a) Una *descripción debe especificar* un atributo no vacío, en el atributo “location” del elemento wsdl:import

2.9.3.4- WSDL y la semántica del atributo location a importar

a) Un *consumidor puede*, pero no necesita, tomar la descripción WSDL de la URI especificada en el atributo “location” de un elemento wsdl:import.

2.9.3.5- Ubicación de los elementos import en el WSDL

a) Cuando aparecen en una *descripción*, los elementos wsdl: import *deben preceder* todos los demás elementos del WSDL namespace excepto los wsdl: documentation.

b) Cuando aparecen en una DESCRIPCIÓN, los elementos wsdl: types DEBEN PRECEDER todos los demás elementos del WSDL namespace excepto los wsdl:documentation y los wsdl:import.

2.9.3.6- Requerimientos de versión del XML

a) Una *descripción debe utilizar* la versión XML 1.0, recomendada por la W3C.

Para la interoperabilidad, los documentos WSDL y los esquemas que importan, expresados en XML, deben usar versión 1.0.

2.9.3.7- Declaraciones del XML namespace

a) Una *descripción no debería contener*, la declaración del namespace xmlns: xml = "http://www.w3.org/XML/1998/namespace".

2.9.3.8- WSDL y el Unicode BOM

a) Una *descripción puede incluir* al Unicode Byte Order Mark (BOM).

XML 1.0 le permite a los documentos que usan la codificación de carácter UTF-8 incluir un BOM.

2.9.3.9- Aceptaciones para la codificación de los datos en el WSDL

a) Una *descripción debe usar* ya sea codificación UTF-8 o UTF-16.

El Perfil consistentemente requiere ya sea UTF-8 ó codificación UTF-16 para ambos, SOAP y WSDL.

2.9.3.10- Coacción Namespace

a) El atributo "targetNamespace" en el elemento wsdl: definitions de una descripción que está siendo importada *debe tener* de la misma forma el valor, como el atributo "namespace" en el elemento wsdl: import en la *descripción* importadora.

2.9.3.11- WSDL y el elemento documentation

a) En una *descripción* el elemento wsdl: documentation *puede* estar presente como el primer hijo de los elementos wsdl: import, wsdl:part y wsdl:definitions además de los elementos a los que se hace referencia en la especificación WSDL 1.1.

2.9.3.12- WSDL Extensiones

a) Una *descripción* conteniendo extensiones WSDL, *no las deben utilizar* para contradecir otros requisitos del Perfil.

b) Una *descripción no debería incluir* los elementos de extensión con un atributo wsdl: required con valor "verdadero" en cualquier construcción WSDL (wsdl: binding, wsdl: portType, wsdl:message, wsdl:types o wsdl:import) eso reclama conformidad para el Perfil.

c) Si durante el procesamiento de una descripción, un consumidor encuentra un elemento de extensión en el WSDL que tiene un atributo wsdl: required con un valor boolean de "verdadero", que el consumidor no entienda o no pueda procesar, el *consumidor debe fallar* el procesamiento.

2.9.4- Tipos

El elemento wsdl: types de WSDL 1.1 incluye definiciones de tipo de datos que tienen importancia para el servicio Web descrito. El Perfil coloca las siguientes restricciones relacionado con esas porciones del contenido del elemento wsdl: types que es referido por elementos WSDL que hacen el Perfil tome conformidad:

2.9.4.1- Referencias QName

a) Una *descripción no debe usar* referencias QName para componentes WSDL en namespaces que no han sido ni importados, ni definidos en el documento referente al WSDL.

b) Una referencia QName para un componente del Esquema en una *descripción debe usar* el namespace definido en el atributo "targetNamespace" en el elemento xsd: schema, o para un namespace definido en el atributo "namespace" en un elemento xsd: import dentro del elemento xsd: schema.

Un Esquema XML requiere que cada referencia QName utilice ya sea el targetnamespace, o un namespace importado (uno marcado explícitamente con un elemento xsd: import). Las referencias QName para namespaces representados sólo por importaciones anidadas no están permitidas.

2.9.4.2- Esquemas y la estructura del targetNamespace

a) Todo los elementos xsd: schema contenidos en un elemento wsdl: types de una *descripción* *debe tener* un atributo "targetNamespace" con un valor válido y no nulo, *a menos que* el elemento xsd: schema tenga elemento(s) xsd: import y / o xsd: annotation como sus únicos hijos.

Requiriendo un targetNamespace en todo los elementos xsd: schema que son hijos de wsdl:types es una buena práctica, coloca una carga mínima en los autores de documentos WSDL y evita los casos que no están tan claramente definidos, como podrían ser.

2.9.4.3- soapenc: Array

a) En una *descripción*, las declaraciones *no deben* extender o restringir el tipo soapenc: Array.

b) En una *descripción*, las declaraciones *no deben usar* el atributo wsdl: arrayType en la declaración de los tipos.

c) En una *descripción*, los elementos *no deberían ser llamados* utilizando la convención ArrayOfXXX.

d) Un *envelope* *no debe incluir* el atributo soapenc: arrayType

2.9.4.4- Definiciones del Target Namespace en el WSDL y el Esquema

a) El "targetnamespace" para definiciones WSDL y el "targetnamespace" para definiciones del Esquema, en una *descripción* *pueden ser* el mismo.

Los nombres definidos por los esquemas y los nombres asignados a las definiciones WSDL están en espacios simbólicos separados.

2.9.4.5- Múltiples definiciones GED con el mismo Qname

a) Una *descripción* *no debería contener* múltiples declaraciones de elemento(s) global(s), que comparten el mismo nombre calificado.

Los componentes esquema de todos los hijos xs: schema, y sus importaciones e inclusiones, del elemento wsdl: types comprende un espacio simbólico único, conteniendo todas las declaraciones globales del elemento. Así, cuando las declaraciones globales del elemento comparten un nombre calificado, un componente único será representado en el espacio del símbolo. Si dos declaraciones son idénticas, no

hay ambigüedad en la estructura del componente, pero si las declaraciones difieren, es indeterminado en lo que se refiere a cuál de las declaraciones será representada, lo cual puede conducir a los problemas de interoperabilidad.

2.9.4.6- Múltiples definiciones de Tipos con el mismo QName

a) Una *descripción no debería contener* múltiples definiciones de tipos que compartan el mismo nombre calificado.

2.9.5- Messages

En WSDL 1.1, los elementos wsdl: message son usados para representar definiciones abstractas de los datos siendo transmitidos. Se usa elementos wsdl: binding para definir cómo las definiciones abstractas están obligadas con una serialización específica de mensaje.

Definición: rpc-literal binding

Una "rpc-literal binding" es un elemento wsdl: binding en cuyos hijos, los elementos wsdl: operation, están todas las operaciones en rpc-literal.

Una "rpc-literal operation" es un elemento hijo wsdl: operation de wsdl: binding, cuyos elementos descendientes wsoap12:body especifican el uso del atributo con el valor "literal", y ya sea:

1- El atributo style con el valor "rpc" es especificado en el elemento hijo wsoap12: operation; O

2- El atributo style no esta presente en el hijo del elemento wsoap12: operation, y el elemento wsoap12: binding enmarcado en el wsdl: binding especifica el atributo style con el valor "rpc".

Definición: document-literal binding

Un "document-literal binding" es un elemento wsdl: binding en cuyos hijos, los elementos wsdl:operation, están todas las operaciones en document-literal .

Una "document-literal operation" es un elemento hijo wsdl: operation de wsdl: binding cuyos elementos descendientes wsoap12: body especifican el uso del atributo con el valor "literal" y, ya sea:

1- El atributo style con el valor "document " es especificado en el elemento hijo wsoap12: operation; O

2- El atributo style no está presente en el elemento hijo wsoap12: operation, y el elemento wsoap12: binding enmarcado en el wsdl: binding especifica el atributo style con el valor "document "; O

3- El atributo style no está presente en ambos elementos hijos, wsoap12: operation y wsoap12:binding enmarcado en el wsdl:binding.

2.9.5.1- Uniones (Bindings) y Partes (Parts)

a) Un “document-literal binding” en una *descripción* *debe*, en cada uno de sus elemento(s) wsoap12: body, liste a lo sumo una parte en el atributo “parts”, si el atributo “parts” es especificado.

b) Si un “document-literal binding” en una *descripción* no especifica el atributo “parts” en un elemento wsoap12: body, el correspondiente abstracto wsdl: message *debe definir* cero o un wsdl: part S.

c) Un wsdl: binding en una *descripción* *puede contener* elemento(s) wsoap12: body que especifiquen que las partes cero, forman el soap: Body .

d) Un “rpc-literal binding” en una *descripción* *debe referir*, en su elemento(s) wsoap12: body, sólo para elemento(s) wsdl:part que han sido definidos usando el atributo “type”.

e) Un *envelope* descrito con un rpc-literal binding *no debe tener* el atributo xsi: nil con un valor "1" o "verdadero" en la parte accessoria.

f) Un wsdl: message en una *descripción* *puede contener* wsdl: part S que usan el atributo “elements” proveyendo esos wsdl: part s que no se les refiere por un wsoap12: body, en un rpc-literal binding.

g) Un document-literal binding en una *descripción* *debe referir*, en cada uno de su elemento(s) wsoap12: body, sólo para elemento(s) wsdl: part que han sido definidos usando el atributo “element”.

h) Un binding en una *descripción* *puede contener* elemento(s) wsoap12: header que se refieran a wsdl: part S en el mismo wsdl: message que están referidos por su elemento(s) wsoap12: body.

i) Un *envelope* *debe contener* exactamente un elemento parte accesoria, para cada uno de los elementos wsdl: part vinculados al correspondiente sobre (envelope) del elemento wsoap12: body.

j) En una descripción doc-literal donde el valor de las partes del atributo wsoap12: body es un string vacío, el *envelope* correspondiente *no debe tener* elementos contenidos en el elemento soap: Body.

k) En una descripción rpc-literal donde el valor de las partes del atributo wsoap12: body es un string vacío, el *envelope* correspondiente *no debe tener* elementos parte accesoria.

2.9.5.2- Uniones (Bindings) y Fallas (Fault)

a) Un wsdl: binding en una *descripción debe referir*, en cada uno de sus elementos wsoap12: header, wsoap12: headerfault y wsoap12: fault, sólo para elemento(s) wsdl: part que han sido definidos usando el atributo “element”.

2.9.5.3- Contenido del elemento portType sin vincular

a) Un wsdl: binding en una *descripción debería vincular* cada wsdl: part de un wsdl:message en el wsdl:portType para el cual se refiere a uno de los wsoap12:body, wsoap12:header, wsoap12:fault o wsoap12:headerfault.

Un portType define un contrato abstracto con un denominado set de operaciones y los mensajes abstractos asociados. Aunque no prohibido, es esperado que cada parte abstracta de los mensajes de input , output y fault especificados en un portType está vinculada al wsoap12:body o wsoap12:header (y así sucesivamente), tan apropiado al usar la unión SOAP como está definido en WSDL 1.1.

2.9.5.4- Declaraciones del elemento part

a) Un wsdl: message en una *descripción* conteniendo un wsdl:part que usa el atributo “element” *debe referir*, en ese atributo, a una declaración global del elemento.

2.9.6- PortTypes

En WSDL 1.1, los elementos wsdl: portType se usan para agrupar un set de operaciones abstractas. El Perfil coloca las siguientes restricciones en conformidad con los elemento(s) wsdl: portType.

2.9.6.1- Ordenando los elementos part

a) El orden de los elementos en el soap: Body de un *envelope debe equivaler* al del wsdl: part en el wsdl: message que lo describe, para cada uno de los elementos wsdl: part vinculados al correspondiente sobre del elemento wsoap12:body.

b) Una *descripción puede usar* el atributo parameterOrder de un elemento wsdl: operation para indicar el valor de regreso y método de firmas como un indicio para codificar generadores.

2.9.6.2- Operaciones Permitidas

a) Una *descripción no debe* usar operaciones de tipo Solicit-Response y Notification en una definición wsdl: portType.

2.9.6.3- Operaciones Distintivas

a) Un wsdl: portType en una *descripción debe tener* operaciones con valores bien definidos para sus atributos “name”.

2.9.6.4- Construcción del atributo parameterOrder

a) Un elemento wsdl: operation hijo de un elemento wsdl: portType en una *descripción debe* ser construido, entonces ese atributo parameterOrder, si está presente, omite a lo sumo a 1 wsdl: part del mensaje de salida.

2.9.6.5- Exclusividad de los atributos types y element

a) Un wsdl: message en una *descripción no debe especificar* ambos atributos “type” y “element” en el mismo wsdl: part.

2.9.7- Uniones (Bindings)

En WSDL 1.1, el elemento wsdl: binding suministra el protocolo concreto y las especificaciones de formato de datos para las operaciones y los mensajes definidos por un wsdl: portType particular. El Perfil coloca las siguientes restricciones en conformidad con las especificaciones binding:

2.9.7.1- Uso de la unión SOAP

a) Un elemento wsdl: binding en una *descripción debe usar* la unión SOAP 1.2 como está definido en el WSDL 1.1 Binding, extensión para SOAP 1.2.

2.9.8- Uniones SOAP (SOAP Bindings)

WSDL 1.1 Binding, extensión para SOAP 1.2, define una unión para los puntos finales de SOAP 1.2. El Perfil promulga el uso de la unión SOAP 1.2 como esta definido el WSDL 1.1 Binding, extensión para SOAP 1.2, y coloca las siguientes restricciones en su uso:

2.9.8.1- Especificando el atributo transport

a) El elemento wsdl: binding en una *descripción debe* ser construido, entonces su elemento hijo wsoap12: binding especifica el atributo “transport”.

2.9.8.2- Transporte HTTP

a) Un elemento wsdl: binding en una *descripción debe especificar* el protocolo de transporte de HTTP con la unión SOAP. Específicamente, el atributo “transport” de su hijo wsoap12: binding *debe tener* el valor "http://schemas.xmlsoap.org/soap/http".

2.9.8.3- Consistencia del atributo style

a) Un wsdl: binding en una *descripción debe ser* una unión rpc-literal o una unión document-literal.

El estilo, "document " o "rpc", de una interacción es especificado en el nivel wsdl:operation, permitiendo wsdl:binding S cuyo wsdl:operation S tienen diferentes style S. Esto ha conducido a los problemas de interoperabilidad. Adicionalmente, el uso de document-literal binding , que generalmente tiene previstas implementaciones más simples que el rpc-literal binding, es promovido. Este indicio no es siempre apropiado, especialmente en el caso de algunas implementaciones existentes.

2.9.8.4- Codificación y el atributo use

El Perfil prohíbe el uso de codificaciones, incluyendo la codificación de SOAP.

a) Un wsdl: binding en una *descripción debe usar* el valor de "literal" para el atributo “use” en todos los elementos wsoap12: body, wsoap12: fault, wsoap12: header y wsoap12: headerfault.

2.9.8.5- Múltiples uniones para elementos portTypes

El Perfil explícitamente permite múltiples uniones (bindings) para el mismo portType.

a) Un wsdl: portType en una *descripción puede* tener cero o más wsdl: binding S que se refieran a él, definidos en el mismo u otros documentos WSDL.

2.9.8.6- Firma de operaciones

a) Las operaciones en un wsdl: binding en una *descripción debe* resultar en firmas de operación que son diferentes una de la otra.

2.9.8.7- Múltiples puertos en un punto final

Cuando los mensajes input destinados para dos wsdl: port S diferentes en el mismo punto final de la red son indistinguibles en el red, no puede ser posible determinar el wsdl:port que está siendo invocado por ellos. Esto puede causar problemas de interoperabilidad. Sin embargo, puede haber situaciones (versiones SOAP, versión de aplicaciones, conformidad para perfiles diferentes) donde gusta localizar más de un puerto (port) en un punto final (endpoint); por consiguiente, el Perfil permite esto.

a) Una *descripción no debería tener* más de un wsdl: port con el mismo valor, para el atributo “location” del elemento wsoap12: address.

2.9.8.8- Elementos hijos para las uniones Document-Literal

a) Un document-literal binding *debe* ser serializado como un *envelope* con un soap: Body cuyo elemento hijo es una instancia de la declaración del elemento global para la que se estableció referencias por la parte correspondiente del wsdl: message.

2.9.8.9- Operaciones One-Way

a) Para operaciones one-way, un *mensaje* de respuesta de HTTP *puede contener* un sobre (envelope).

b) Para operaciones one-way, un *consumidor no debe interpretar* un código exitoso de estatus de respuesta de HTTP (i.e., 2xx), como que signifique que el mensaje es válido, o que el receptor lo pueda procesar.

2.9.8.10- Namespaces para elementos wsoap12

Hay confusión acerca de como el namespace es asociado con los elementos hijos de diversos hijos de soap: Envelope, que ha conducido a las dificultades de interoperabilidad. El Perfil define estos:

a) Un document-literal binding en una *descripción no debe tener* el atributo “namespace” especificado dentro de los elementos wsoap12: body, wsoap12: header wsoap12: headerfault y wsoap12: fault.

b) Un rpc-literal binding en una *descripción debe tener* el atributo “namespace” especificado, el valor del cual *debe ser* un URI absoluto, dentro de los elementos wsoap12: body.

c) Un rpc-literal binding en una *descripción no debe tener* el atributo “namespace” especificado dentro de los elementos wsoap12: header, wsoap12: headerfault y wsoap12: fault.

En un document-literal SOAP binding, el elemento hijo publicado como serial del soap: Body obtiene su namespace del targetNamespace del esquema que define el elemento. El uso del atributo namespace del wsoap12: body pasaría sobre la disposición de namespace del elemento. Esto no es admitido por el Perfil.

Inversamente, en un rpc-literal SOAP binding, el elemento hijo publicado en serie del elemento soap:Body consiste en un elemento de envoltura, cuyo namespace es el valor del atributo namespace del elemento wsoap12:body y cuyo nombre local es ya sea el nombre de la operación o el nombre de la operación añadida como sufijo con "respuesta" (Response). El atributo namespace es requerido, como está opuesto a ser optativo, para asegurar que los hijos del elemento soap: Body son un namespace calificado.

2.9.8.11- Consistencia de los elementos portType y binding

a) Un wsdl: binding en una *descripción debe tener* el mismo set de wsdl: operation S como el wsdl: portType, al cual hace referencia.

2.9.8.12- Enumeración de fallas

Una descripción de servicio Web debería incluir todas las fallas conocidas ha la hora que el servicio esté definido. Existe también la necesidad permitir la generación de fallas nuevas, que no habían estado identificadas cuando el servicio Web fue definido.

a) Un wsdl: binding en una *descripción debería contener* un wsoap12: fault describiendo cada falla conocida.

b) Un wsdl: binding en una *descripción debería contener* un wsoap12: headerfault describiendo cada falla conocida del encabezado.

c) Un *envelope puede contener* fallas con un elemento “detail”, que no se describieron por el elemento wsoap12: fault en la correspondiente descripción WSDL.

d) Un *envelope puede contener* los detalles de un encabezado procesando fallas relacionadas con un bloque del encabezado de SOAP (SOAP header block) que no se describieron por el elemento wsoap12:headerfault en la correspondiente descripción WSDL.

2.9.8.13- Tipos y nombres de los elementos de la unión SOAP

a) Un wsdl:binding en una *descripción debe usar* el atributo “part” con un tipo de esquema "NMTOKEN" dentro de todos los elementos wsoap12:header y wsoap12:headerfault.

b) Un wsdl:binding en una *descripción no debe usar* el atributo “parts” dentro de los elementos wsoap12:header y wsoap12:headerfault.

2.9.8.14- Atributo name en las fallas

a) Un wsdl:binding en una *descripción debe tener* el atributo “name” especificado dentro de todos los elementos wsoap12:fault.

b) En una *descripción*, el valor del atributo “name” en un elemento wsoap12:fault *debe corresponder* al valor del atributo “name” en su elemento padre wsdl:fault.

2.9.8.15- Omisión del atributo use

a) Un wsdl:binding en una *descripción puede especificar* el atributo “use” dentro de los elementos wsoap12:fault.

b) Si en un wsdl:binding en una *descripción* el atributo “use” dentro de un elemento wsoap12:fault está presente, su valor *debe ser* "literal".

2.9.8.16- Defectos del atributo use

a) Un wsdl:binding en una *descripción* que contiene uno o más elementos wsoap12:body, wsoap12:fault, wsoap12:header o wsoap12:headerfault, que no especifican el atributo “use” *deben ser* interpretados como si el valor "literal" ha sido especificado en cada caso.

2.9.8.17- Consistencias de sobres (Envelopes) con descripciones

Estos requisitos especifican que cuando una instancia recibe un sobre que no es conforme a la descripción WSDL, una falla debería ser generada, a menos que la instancia se encargue de procesar el sobre, a pesar de esto.

Como está especificado por el modelo de procesamiento de SOAP:

(a) un "VersionMismatch" faultcode debe ser generado si el namespace del elemento "Envelope" es incorrecto.

(b) un "MustUnderstand" fault debe ser generado si la instancia no comprende un bloque del encabezado de SOAP (SOAP header block) con un valor de "1" para el atributo soap:mustUnderstand.

En todos los demás casos, donde un sobre es inconsistente con su descripción WSDL, una falla con un "Client" faultcode debería ser generada.

a) Si una *instancia* recibe un sobre que es inconsistente con su descripción WSDL, *debería generar* un soap:Fault con un faultcode de "Client", a menos que una falla "MustUnderstand" o "VersionMismatch" sea generada.

b) Si una *instancia* recibe un sobre que es inconsistente con su descripción WSDL, *debe revisar* en busca de fallas "VersionMismatch", "MustUnderstand" y "Client" mirando las condiciones en ese orden.

2.9.8.18- Envolturas de respuesta

a) Un *envelope* descrito con un rpc-literal binding que es una respuesta *debe tener* un elemento de envoltura (wrapper), cuyo nombre es el correspondiente wsdl:operation añadido como sufijo con el string "Response".

2.9.8.19- Partes de acceso

a) Un *envelope* descrito con un rpc-literal binding *debe colocar* los elementos de parte de acceso para los parámetros y el valor de retorno, fuera del namespace.

b) Los elementos de parte de acceso en un *message* descrito con un rpc-literal binding *deben tener* un nombre local del mismo valor como el atributo “name” del elemento wsdl:part correspondiente.

2.9.8.20- Namespace para los hijos de las partes de acceso

a) Un *envelope* descrito con un rpc-literal binding *debe declarar* un namespace competente para los descendientes de los elementos de la parte de acceso, para los parámetros y el valor de regreso, como esta definido por el esquema, en el cual los tipos de la parte de acceso están definidos.

2.9.8.21- Encabezados requeridos

a) Un *envelope debe incluir* todos los wsoap12:header S especificados en un wsdl:input o wsdl:output de un wsdl:operation de un wsdl:binding que lo describe.

2.9.8.22- Permitir encabezados sin describir

Los encabezados son mecanismos de extensibilidad de SOAP. Los encabezados que no están definidos en la descripción WSDL pueden necesitar ser incluidos en los sobres por varias razones.

a) Un *envelope puede contener* bloques del encabezado SOAP (SOAP header blocks) que no se describieron, en el wsdl:binding que lo describe.

b) Un *envelope* conteniendo bloques del encabezado SOAP (SOAP header blocks) que no se describieron en el correcto wsdl:binding *puede tener* el atributo “mustUnderstand” en tales bloques del encabezado SOAP puestos a ' 1’.

2.9.8.23- Ordenando encabezados

a) El orden de los elementos wsoap12:header en las secciones wsoap12:binding de una *descripción deben ser* consideradas independientes del orden de los bloques del encabezado SOAP (SOAP header blocks) en el sobre.

b) Un *envelope puede contener* más que una instancia de cada bloque del encabezado SOAP para cada elemento wsoap12:header en el hijo apropiado de wsoap12:binding en la descripción correspondiente.

2.9.8.24- Describiendo el parámetro action en el Encabezado Content-Type MIME

a) Si el parámetro “action” en el encabezado Content-Type está presente en un *message* HTTP, su valor *debe ser igual* al valor correspondiente del atributo soapAction del wsoap12:operation.

2.9.8.25- Encabezado HTTP del SOAPAction

a) Un *receptor no debe confiar* en la presencia de encabezado SOAPAction HTTP para procesar correctamente el mensaje.

b) Un *remitente no debería incluir* el encabezado SOAPAction HTTP.

2.9.8.26- Extensiones de la unión SOAP

El atributo wsdl:required ha sido ampliamente incomprendido y usado por autores de WSDL, algunas veces para indicar incorrectamente la opcionalidad de wsoap12:header S. El atributo wsdl:required, como está especificado en WSDL 1.1, es un mecanismo de extensibilidad apuntado a procesadores WSDL. Deja elementos nuevos de la extensión WSDL para ser introducidos en una manera simple. El uso de wsdl:required es para señalar al procesador WSDL que el elemento de la extensión necesita ser reconocido y comprendido por el procesador WSDL para que la descripción WSDL sea correctamente tratada.

a) Un *consumidor debe comprender* y *debe procesar* todos los elementos de la extensión de WSDL 1.1 SOAP Binding, sin distinción de la presencia o la ausencia del atributo wsdl:required en un elemento de extensión; y sin distinción del valor del atributo wsdl:required, cuando está presente.

b) Un *consumidor no debe interpretar* la presencia del atributo wsdl:required en un elemento de extensión wsoap12 con un valor de "falso", para querer decir que el elemento de extensión es optativo, en los sobres generados de la descripción WSDL. [14]

Conclusiones:

Este capitulo fue enfocado hacia la WS-I (Web Service Interoperability), que no es mas que la organización que rige la interoperabilidad de los servicios web. Se trataron además los diferentes perfiles y

patrones que esta estipula, haciendo énfasis en el perfil básico, así como las diferentes restricciones y especificaciones que se deber tener en cuenta para su adecuado uso.

CAPÍTULO III

Propuesta del Procedimiento

Procedimiento para el proceso de elaboración de las descripciones de los servicios Web (wsdl) del Sistema de Información para la Salud, siguiendo los estándares definidos por la WS-I.

Desde el punto de vista informático, interoperabilidad se define como la habilidad que tiene un sistema o producto para trabajar con otros sistemas o productos sin un esfuerzo especial por parte del cliente. Donde más se ve realmente la aplicación de este concepto es en la materialización de los servicios Web en la red.

3.1 Objetivo

Lograr una correcta definición para la elaboración de los wsdl, cumpliendo con las pautas seguidas por el perfil básico, propuesto por la WS-I, ya que los mismos representan la base para el buen manejo de la interoperabilidad en los servicios Web.

3.2 Alcance

Enmarcado en el proceso de elaboración de las descripciones de los servicios Web (wsdl) del Proyecto Productivo APS.

3.3 Acrónimos

PB: Perfil Básico planteado por la WS-I.

3.4 Descripción

Durante la elaboración de las descripciones de los servicios Web (wsdl), se han de tener en cuenta una serie de pasos y especificaciones para que el futuro servicio Web no presente problemas de interoperabilidad, brindándose mayores posibilidades a los futuros clientes a que consuman el servicio Web, sin importar la plataforma o el lenguaje de programación usado.

Los **estándares propuestos** para el desarrollo del procedimiento son los siguientes:

1. Uso de XML para la codificación.
2. Uso de SOAP para el formato de los mensajes.
3. Uso de WSDL para la descripción del servicio.
4. Uso de UDDI para el descubrimiento del servicio.

La explicación sobre los estándares antes mencionados, recordar que están en el Capitulo I → **1.4.3 Estándares empleados pag** 12 .

Entrando en detalles para la propuesta de procedimiento, plantear que para el proceso de elaboración de las descripciones (wsdl), se usó una herramienta de manera particular, por su fácil aprendizaje y las posibilidades que brinda:

La herramienta, *Altova XMLSpy 2007 Enterprise Edition*, la cual es la líder del mercado en el entorno de desarrollo XML, proporcionando intuitivas vistas y potentes utilidades para modelar, editar, transformar y depurar tecnologías relacionadas con XML de forma rápida y fácil. Su avanzado manejo de errores y sus capacidades de hiperenlace dinámico en el validador XML hacen de la corrección de documentos XML una tarea sencilla.

3.5- Procedimiento a seguir para realizar las descripciones del servicio Web (wsdl)

Para llevar a cabo el desarrollo del procedimiento se ha de tener en cuenta un orden lógico en los distintos pasos a seguir, descritos a continuación:

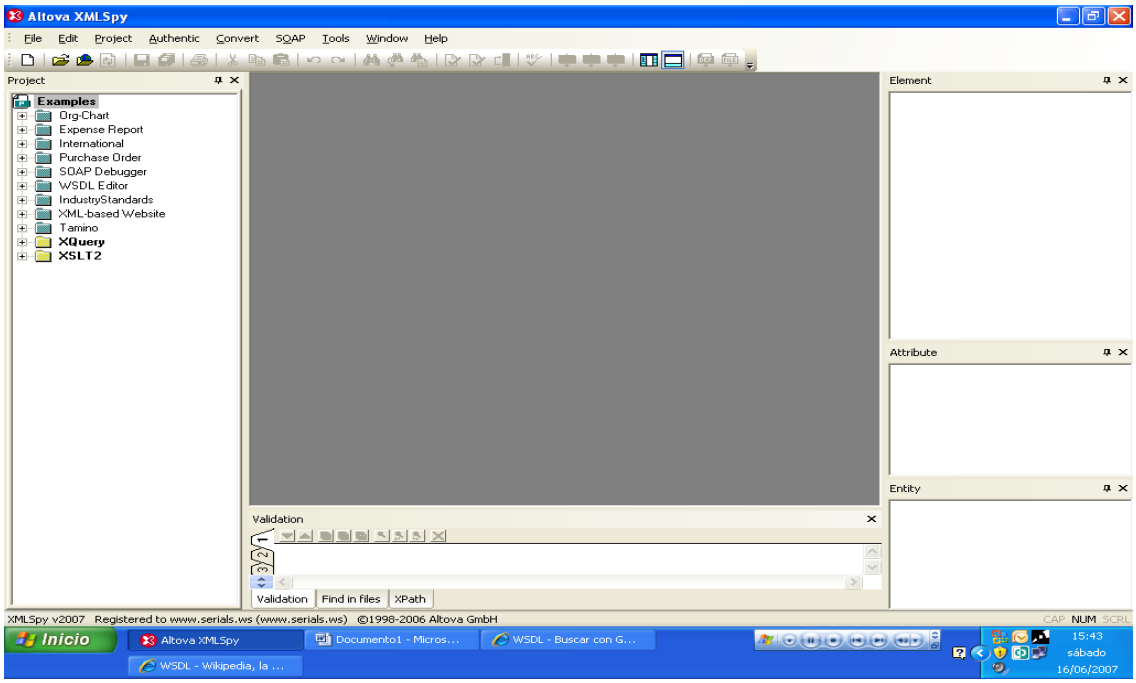
1- Creación del documento de extensión wsdl.
2- Especificaciones a tener en cuenta en el elemento *portType*.
3- Especificaciones a tener en cuenta en el elemento *bindings*.
4- Especificaciones a tener en cuenta en el elemento *service*.
5- Verificación de la validez y la correcta estructura de la descripción.
6- Realización de pruebas contra el servidor.

Los autores se dieron a la tarea de elaborar una plantilla la cual ponen a su disposición (**Template.wsdl**), la misma se realizó para lograr una estandarización en las descripciones realizadas en el Proyecto productivo APS, en ella se definen, para cada una de las partes de un WSDL, las normas y estándares a seguir, acorde a las definiciones/especificaciones planteadas por el Perfil Básico definido por la WS-I. El objetivo es servir de ejemplo, para la elaboración correcta de las descripciones, evitando errores que se puedan cometer, puesto que los aspectos básicos que se debían tener en cuenta para cumplir con el Perfil Básico, ya están prácticamente solucionadas en su totalidad en la plantilla, solamente es continuar el desarrollo de la descripción teniendo es cuenta los aspectos especificados en ella, que son de estricto cumplimiento para dar conformidad con lo planteado en el Perfil Básico.

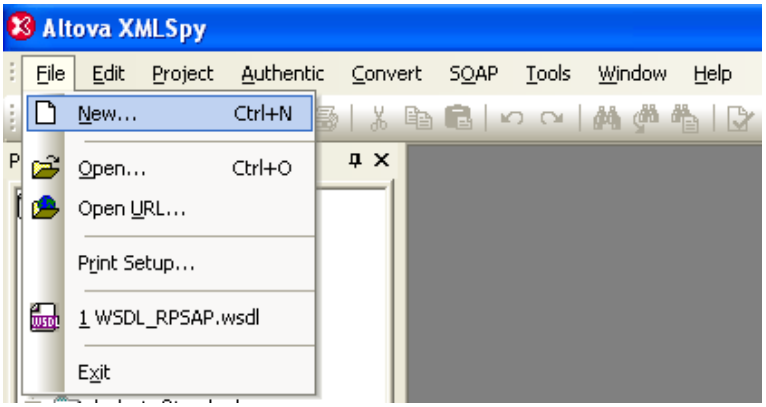
Es válido aclarar, que los nombres definidos para cada uno de los elementos y atributos de la descripción, no están especificados en el PB, se realizó con la idea de lograr una estandarización, para un trabajo más organizado con las descripciones de los servicios Web, dentro del marco de trabajo del Proyecto productivo APS.

3.5.1- Creación del documento de extensión wsdl

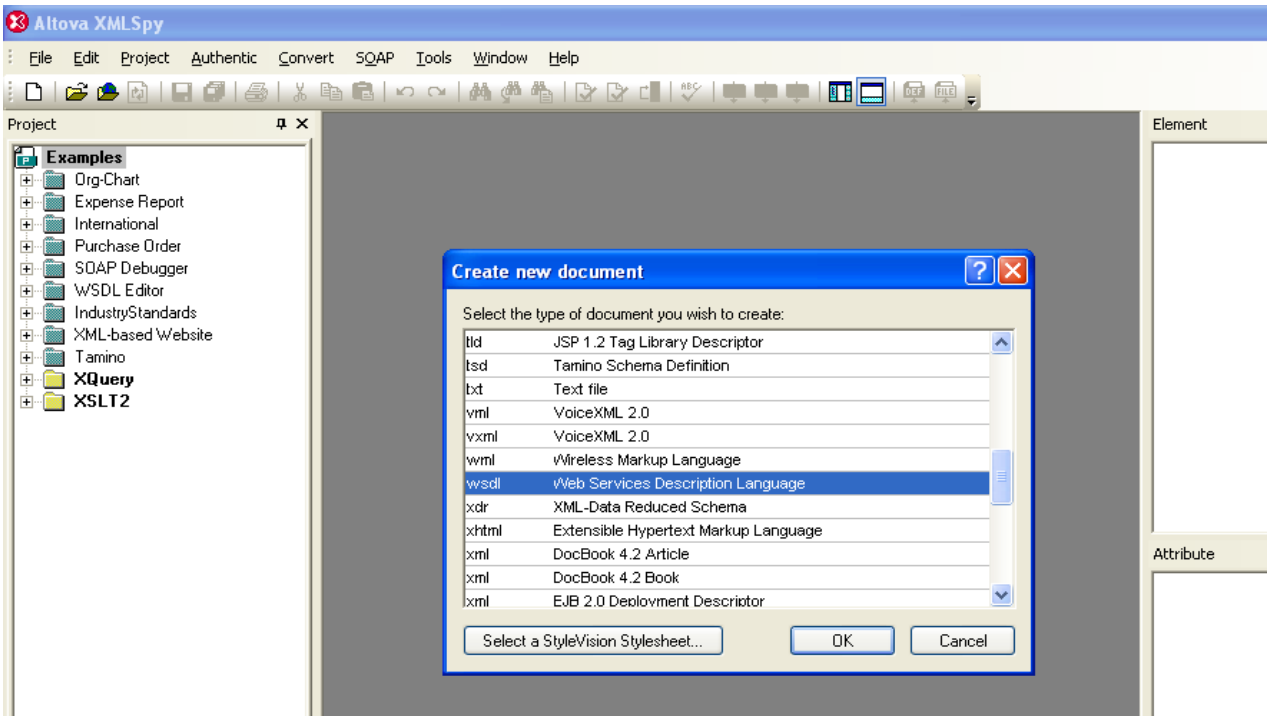
Una vez abierto el Altova XMLSpy 2007 se tendría una ventana como la siguiente:



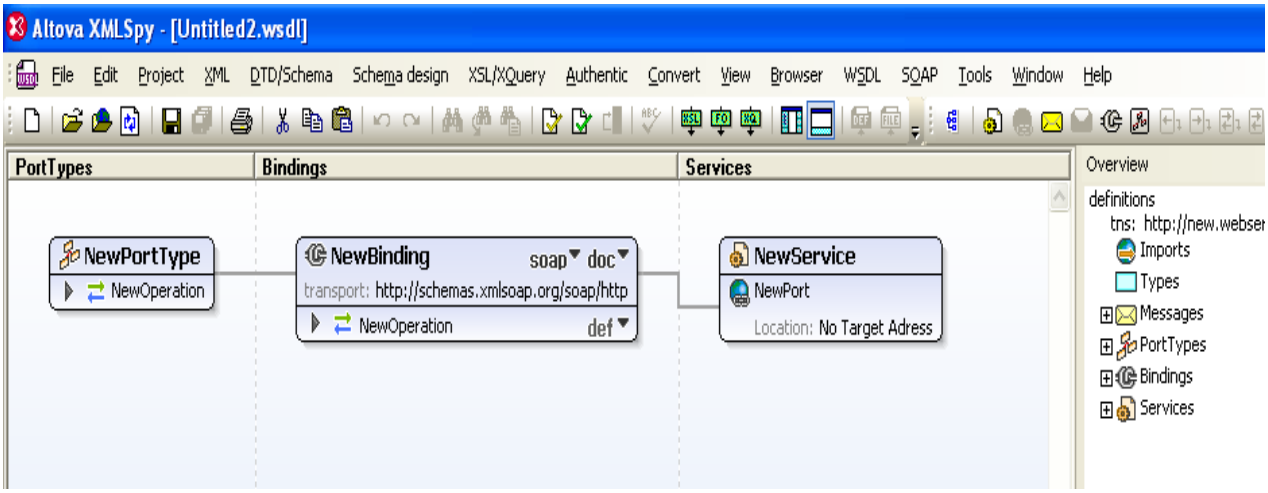
Luego se va a la barra de menús, específicamente **File** y se da clic en **New**



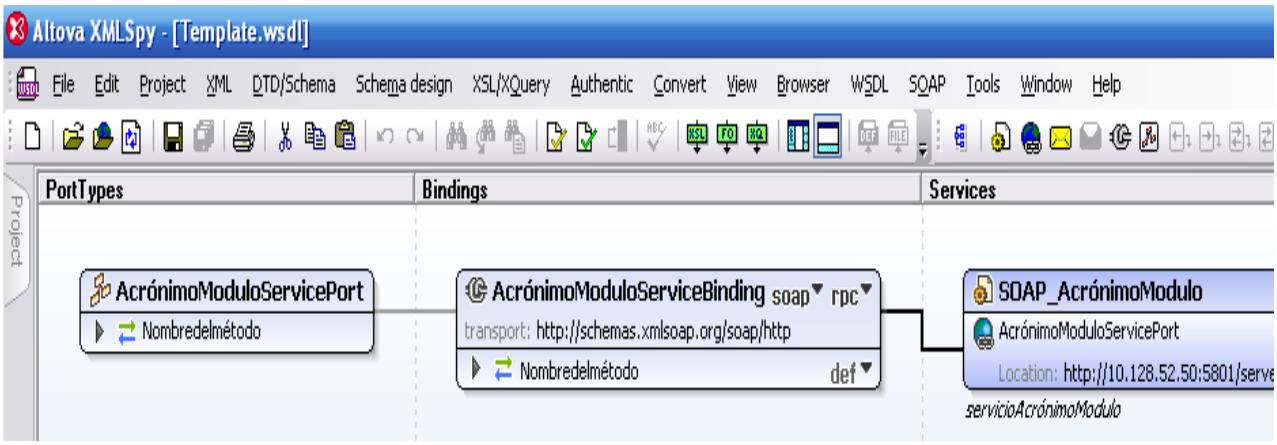
Se muestra una ventana como la que a continuación se ve, de la cual se selecciona la opción crear un documento de tipo **wsdl**.



Obteniéndose la siguiente estructura:



Llegado este punto y para evitar errores a la hora de realizar la mayoría de los cambios necesarios, para dar cierta conformidad con el Perfil Básico, se propone el uso de la plantilla **Template.wsdl**:



En la misma ya han sido chequeados aspectos definidos por el Perfil Básico, como es el caso del uso de *xml***,** la *codificación* UTF-8 ó UTF- 16 y los distintos *namespace* para cada prefijo, cumpliendo con las especificaciones:

2.9.1- Verificación de los distintos namespace en el documento.

2.9.3.6- Requerimientos de versión del XML.

2.9.3.9- Aceptaciones para la codificación de los datos en el WSDL.

Planteadas dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

Ejemplo código:

```
<?xml version="1.0" encoding="UTF-8"?>
 <!-- edited with XMLSpy v2007 (http://www.altova.com) by www.serials.ws (www.serials.ws) -->
]<wsdl:definitions
xmlns:tns="urn:servicioAcrónimoModulo"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
targetNamespace="urn:servicioAcrónimoModulo">
```

Explicar que estos no son los únicos requisitos que se han tenido en cuenta, para dar la conformidad con el PB, respecto a la elaboración de las descripciones, más a adelante en el transcurso del documento se hará énfasis en otros que han sido tratados.

3.5.2- Especificaciones a tener en cuenta en el elemento *portType*

Todos los wsdl deben contar con un solo puerto. Este puerto debe llamarse "Acrónimo del Módulo+ServicePort".

Ejemplo código:

```
<portType name="RLServicePort">
```

Debe incluir una operación por cada método php al que se enlazará posteriormente, y deben coincidir los nombres de la operación y el método php. El prefijo de cada operación debe ser "wsdl:" y el de los nombres de los mensajes "tns:"

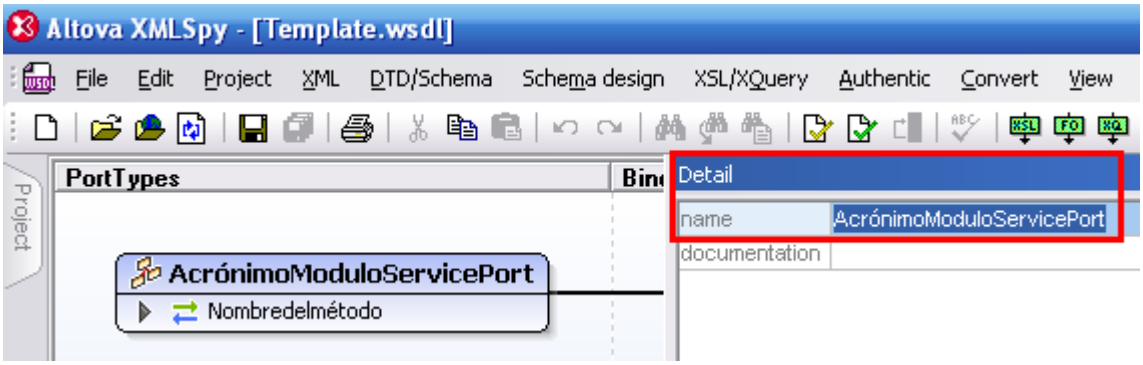
Ejemplo código:

```
<wsdl:operation name="BuscarTuplaZona">
<wsdl:documentation>Este método retorna ...</wsdl:documentation>
<wsdl:input message="tns:BuscarTuplaZonaResponse"/>
<wsdl:output message="tns:BuscarTuplaZonaRequest"/>
</wsdl:operation>
```

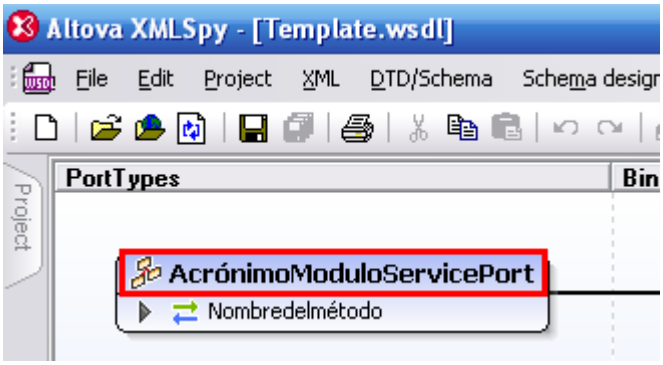
Nota: el método php se llama BuscarTuplaZona.

Para cambiar el nombre del **portType**, hay que situarse donde dice AcrónimoMóduloServicePort, poniendo el nombre correspondiente como se indica, por ejemplo si el módulo es RAS, el nombre quedaría RASServicePort, esto se puede hacer dando doble clic encima de AcrónimoMóduloServicePort, o cambiándolo en el name dentro de *Detail* (se tiene que haber dado clic encima de AcrónimoMóduloServicePort) como se muestra a continuación:

Cambiando el nombre en el atributo name dentro de *Detail*:



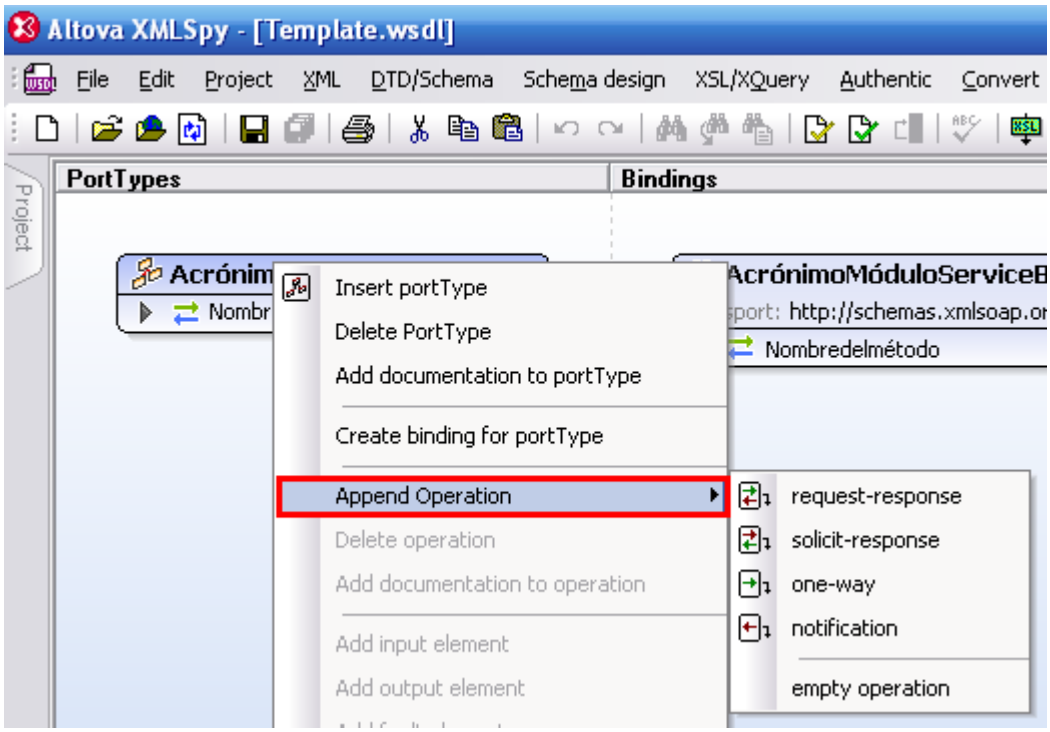
Cambiando el nombre directamente sobre AcrónimoMóduloServicePort:



3.5.2.1- Agregando nuevas operaciones

a) Clic derecho encima del nombre del **portType** que en este ejemplo como tal sería AcrónimoMóduloServicePort**.**

b) Ir a *Append Operation*, ya una vez aquí escoger el tipo de operación que se ajuste a lo que hace el método con el que está trabajando.



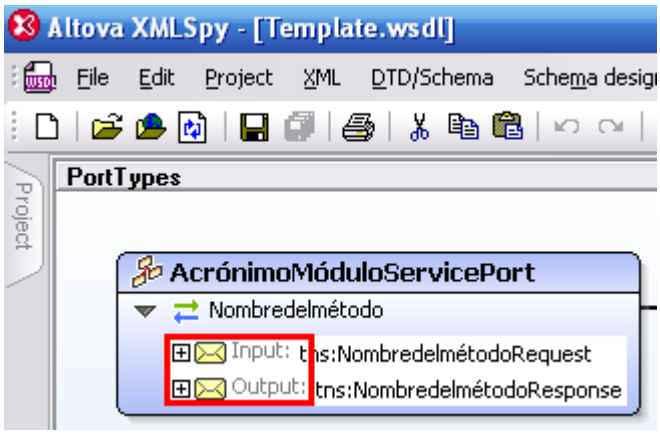
request-response: Los métodos que realizan una consulta pasando parámetros de entrada, y esperan una respuesta con parámetros de salida. Ejemplo: métodos de búsquedas

solicit-response: En este tipo de operación el servicio realiza una solicitud y espera parámetros de respuesta.

one-way: Los métodos que realizan una consulta pasando parámetros de entrada y no reciben respuesta.

notification: El Servicio envía un mensaje, y no recibe respuesta.

Si el tipo de operación es de *request-response*, la operación tendría parámetros de entrada (Input) y de salida (Output):



En cualquiera de los otros casos solo tendrían parámetros de entrada (Input) o salida (Output) según el caso.

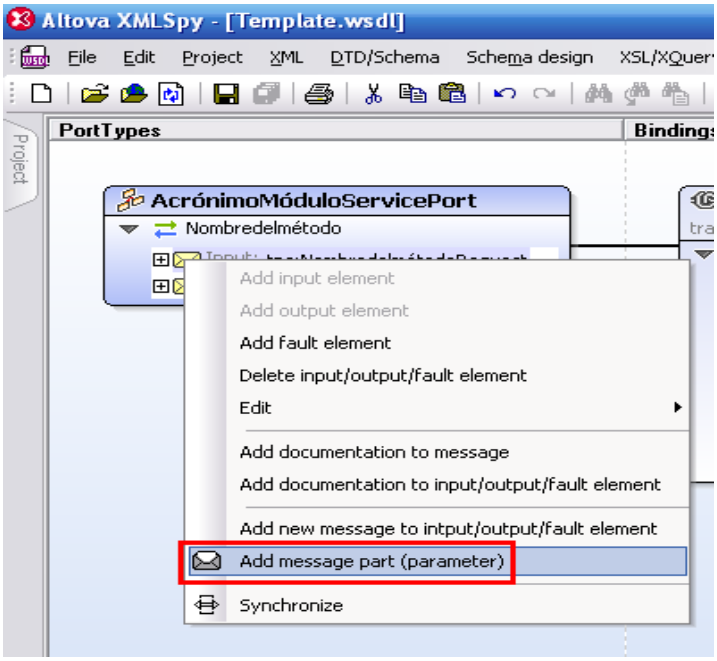
Según el PB, no se debe hacer uso de las operaciones de tipo *solicit- response*, ni *notification*, se debe definir bien el *nombre de las operaciones*, aclarar además que un *portType* puede tener asociado múltiples *bindings*, estos planteamientos quedan expuestos respectivamente, en las especificaciones:

2.9.6.2- Operaciones Permitidas.
2.9.6.3- Operaciones Distintivas.
2.9.8.5- Múltiples uniones para elementos portTypes.

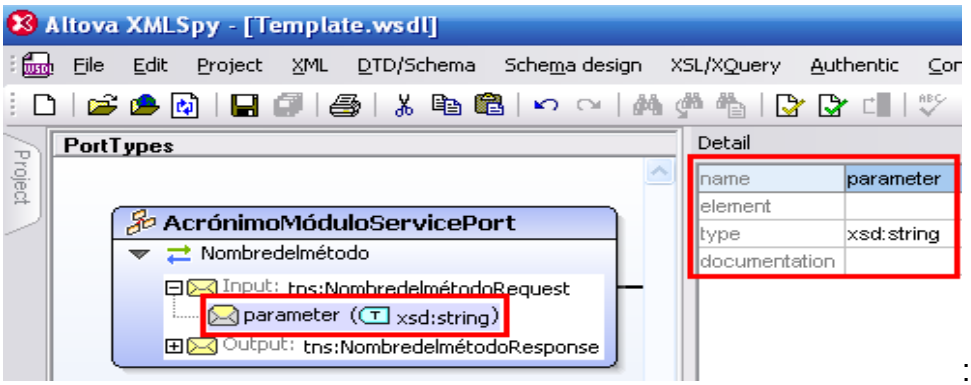
Planteadas dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

3.5.2.2- Agregando parámetros a las operaciones

Para agregar los parámetros de entrada y/o salida se da clic derecho encima del tipo de parámetro que se va a agregar (Input/Output) y se selecciona *Add Message Part (Parameter)*, agregándose tantos como se necesite, según la cantidad en el método php:



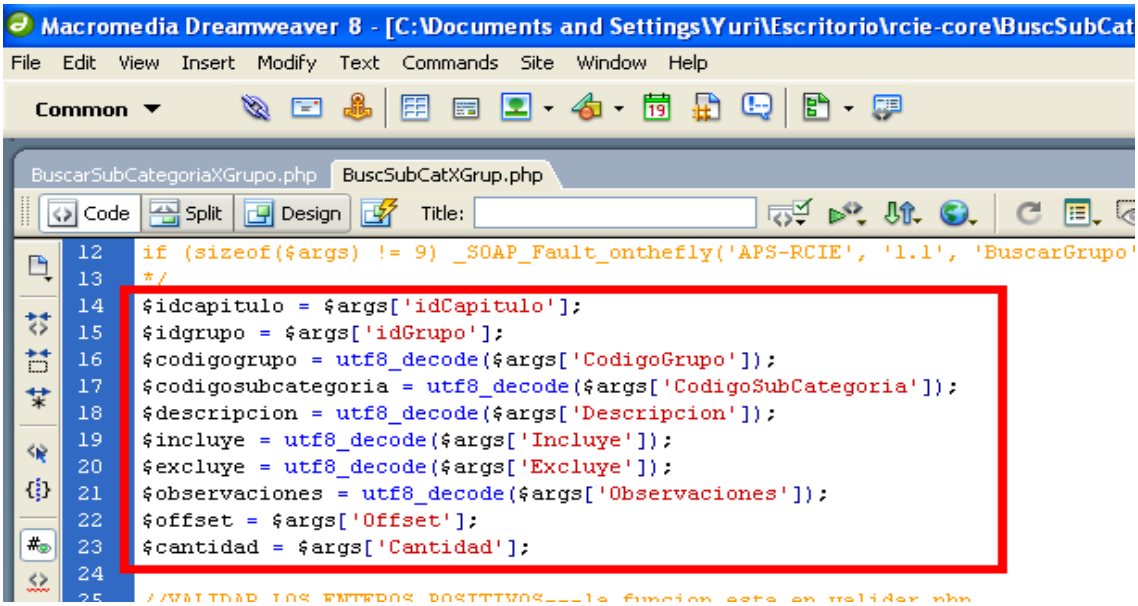
Todos los parámetros tienen un tipo específico (string, int, double, etc.), cualquiera de los tipos definidos en el XMLSchema o uno de los definidos por usted, se modifica dando doble clic encima del parámetro que se pasa, o en *Detail*, lo mismo sucede con el nombre del parámetro:



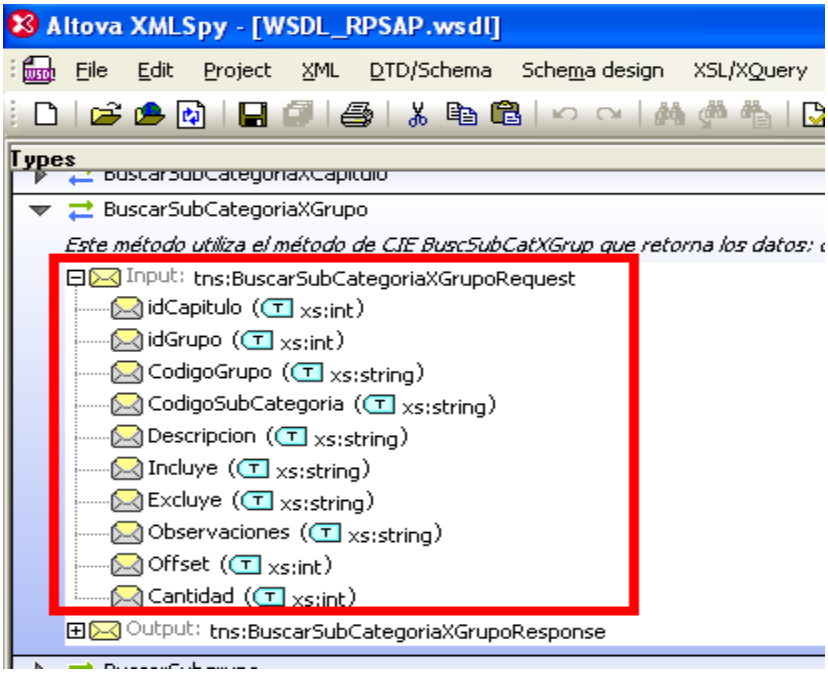
Es importante tener en cuenta que los parámetros deben estar en el mismo orden en el que son definidos en el método php, al cual hace referencia la operación y deben coincidir los nombres y los tipos de datos.

Ejemplo de cómo quedarían los parámetros tomados desde la un método php:

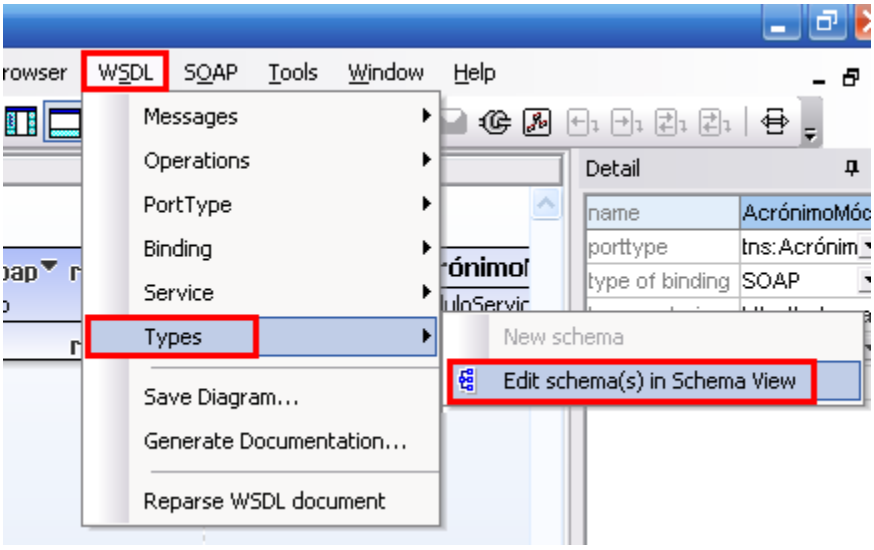
Tomando los parámetros de entrada de la Pág. PHP:



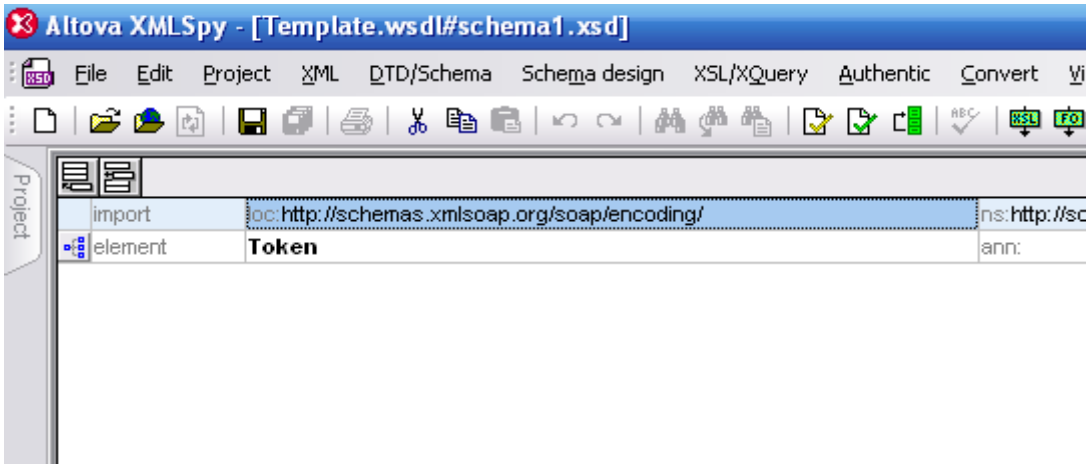
Al conformar el WDSL quedaría compuesto de la siguiente manera (Nótese que coinciden los nombres y los tipos):



Ahora bien los parámetros de salida (Output), por lo general devuelven un mensaje soap que es un arreglo, y para los mismos hay que definir tipos de datos complejos, para ello se debe ir a la vista de esquemas:



Como bien se muestra en la figura anterior, primero se va en la barra de menús a **WSDL**, luego a *Types*, y por último a *Edit schema(s) in Schema View*, se da aceptar en la ventana de diálogo que sale, obteniéndose la siguiente vista:



Aclarar que el elemento **Token**, aparece porque se había definido en la plantilla, de lo contrario no estuviese. También se puede llegar hasta la ventana anterior, dando clic en el icono  que aparece en la barra de herramientas.

3.5.2.3- Creando tipos de datos complejos

Como bien se plantea los arreglos se deben definir como tipos de datos complejos en una descripción (wsdl). Si a algún método en php se le pasa como parámetro un arreglo o devuelve un arreglo, este arreglo se representa en una descripción (wsdl) como un tipo de dato complejo.

En estos tipos de datos complejos el nombre debe comenzar con la palabra "ArregloResultado" y posteriormente un nombre que identifique el arreglo, este nombre debe coincidir con el nombre del método php para el cual se está realizando el tipo de dato complejo. Aclarar que anteriormente se ponía el nombre "ArrayResultado" en las descripciones pero el PB plantea que no se debe comenzar los nombres con la palabra *array*, ver especificación:

2.9.4.3- soapenc: Array inciso c)

Planteada dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

Ejemplo código:

```
<complexType name="ArregloResultadoBuscarTuplaZona">
  <sequence>
    <element name="TuplaZona" type="tns:StructTuplaZona"/>
  </sequence>
</complexType>
```

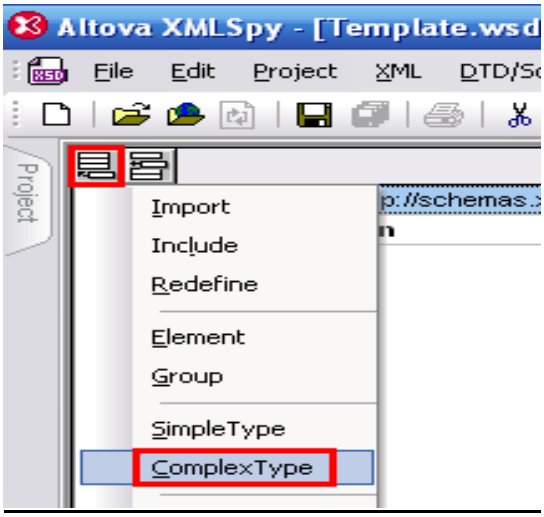
Para construir la estructura interna de un arreglo se escogerá el modelo “sequence” antes de definir los elementos del arreglo y se le pondrá como nombre “Struct+nombre de la estructura”, donde el nombre de la estructura coincidirá el nombre que identifica el arreglo. Cada elemento del arreglo tiene un nombre y un tipo de dato (que tienen que coincidir con el nombre de ese elemento y el tipo de dato esperado en el arreglo de salida del método php).

Ejemplo código:

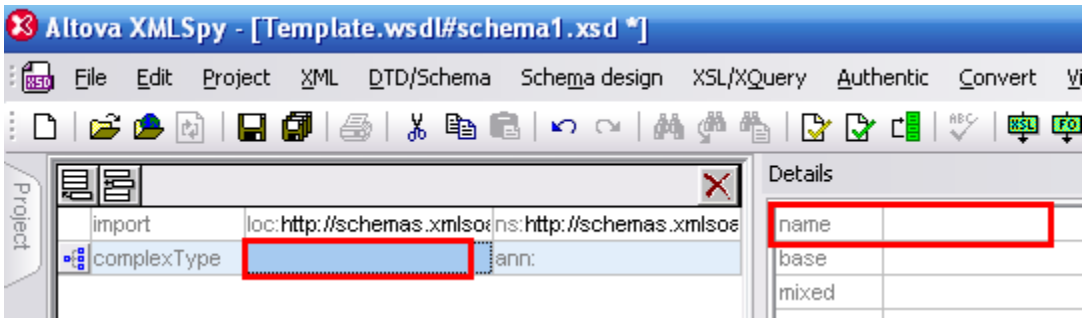
```
<complexType name="StructTuplaZona">
   <sequence>
     <element name="nom_zona" type="string"/>
     <element name="id_circunscrip" type="int"/>
     <element name="id_consejo" type="int"/>
   </sequence>
</complexType>
```

Los tipos de datos así creados se deben describir, explicando el significado del arreglo como tal y de cada uno de sus atributos; especificando el tipo de datos de cada atributo.

Para hacer el arreglo como tal, se da clic izquierdo en el botón marcado a continuación, y se selecciona *Complex Type*

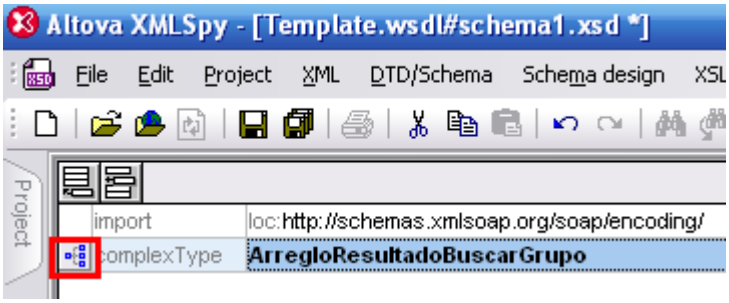


Obteniéndose una ventana como la que se muestra a continuación:

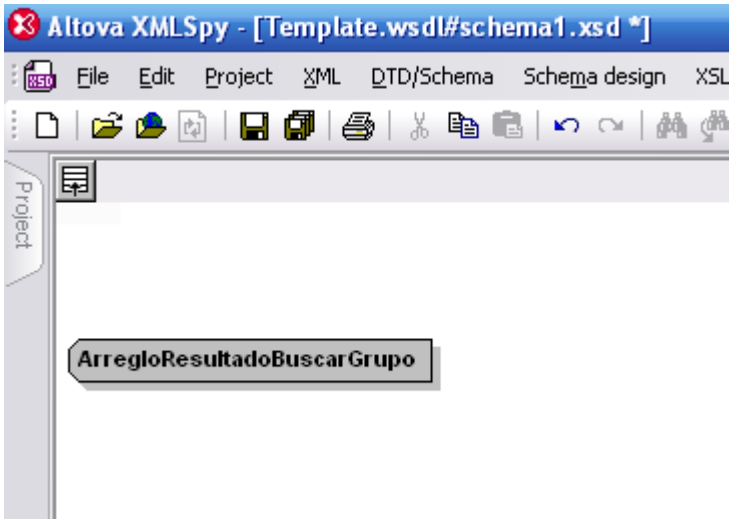


El nombre del arreglo se cambia dando doble clic en la barra azul donde esta marcado el rectángulo o en *Details*, debe comenzar con la palabra "ArregloResultado" y posteriormente un nombre que identifique el arreglo, este nombre debe coincidir con el nombre del método php, para el cual se está realizando el tipo de dato complejo, como se planteaba anteriormente.

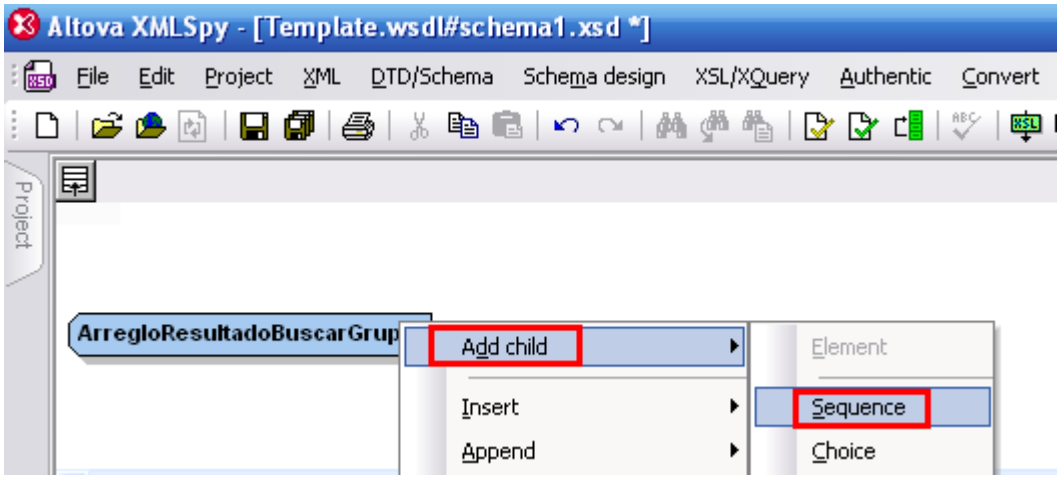
Para crear la estructura interna del arreglo o tipo complejo, se da clic en el botón marcado a continuación:



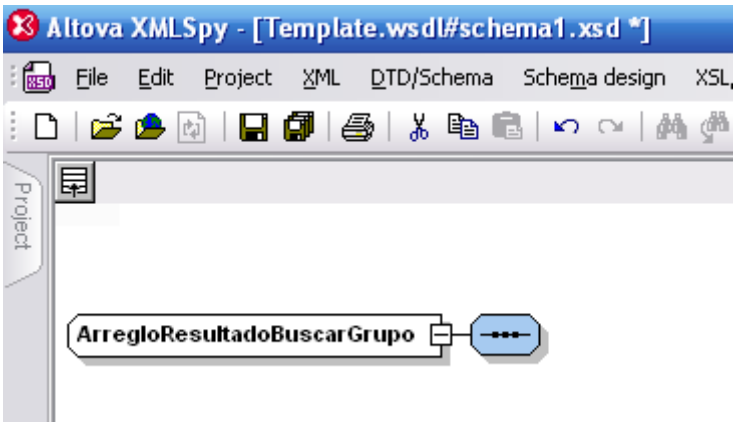
Una vez realizado este paso se vería lo siguiente:



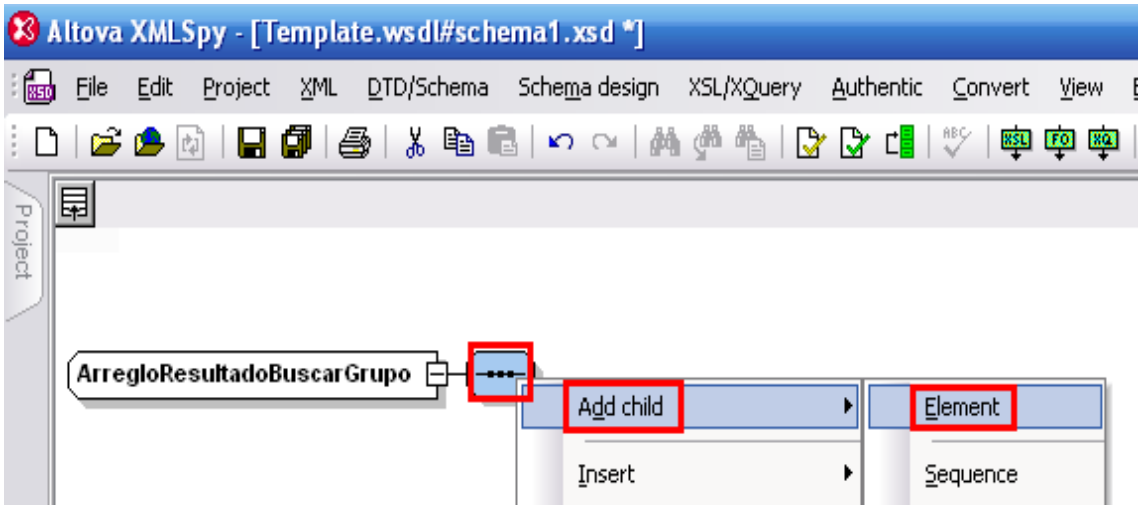
Ahora ya se está en condiciones para crear la estructura interna del arreglo, para ello se da clic derecho encima de ArregloResultado + Nombre método (En el ejemplo ArregloResultadoBuscarGrupo), se selecciona *Add child* y de ahí se escoge *Sequence* (Es el que más se utiliza).



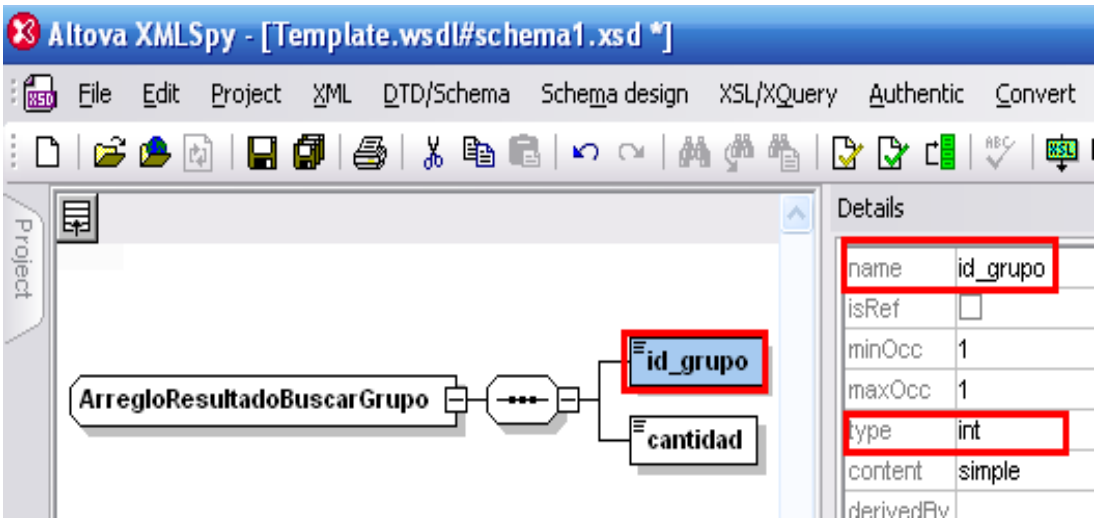
Quedando de la siguiente manera:



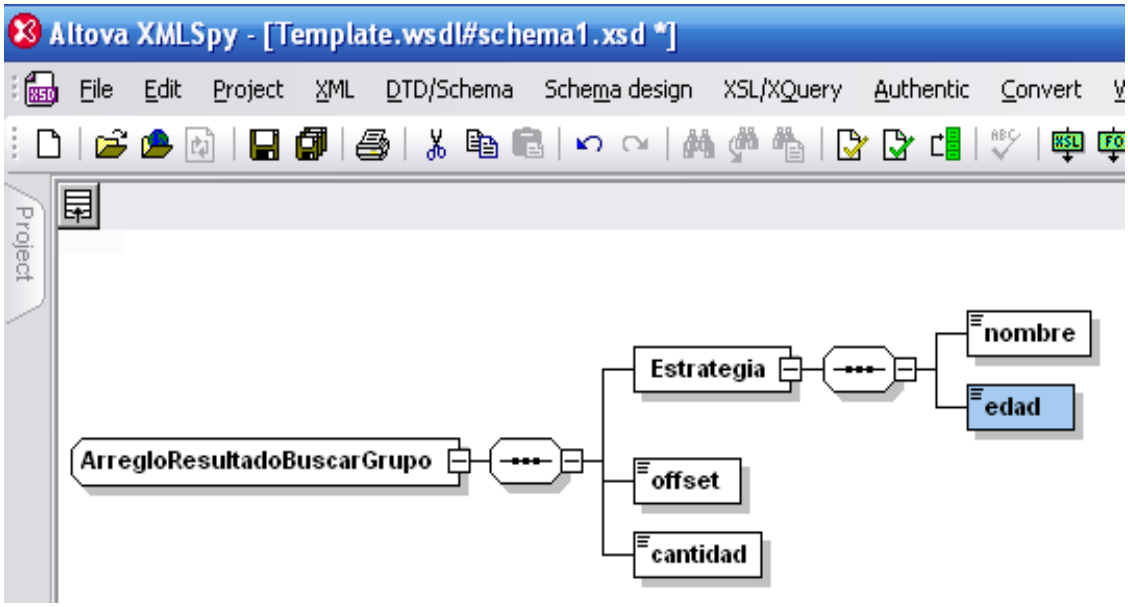
Se da clic derecho en la nueva parte agregada y se selecciona *Add child* y luego *Element*, se añade tanto elementos como contenga el arreglo que devuelva el método:



Los nombres de los elementos deben coincidir con los que contiene el arreglo que se devuelve en el método implementado, al igual que el tipo (string, int, double), y estos se pueden modificar donde está marcado en la figura, en la ventana *Details*:

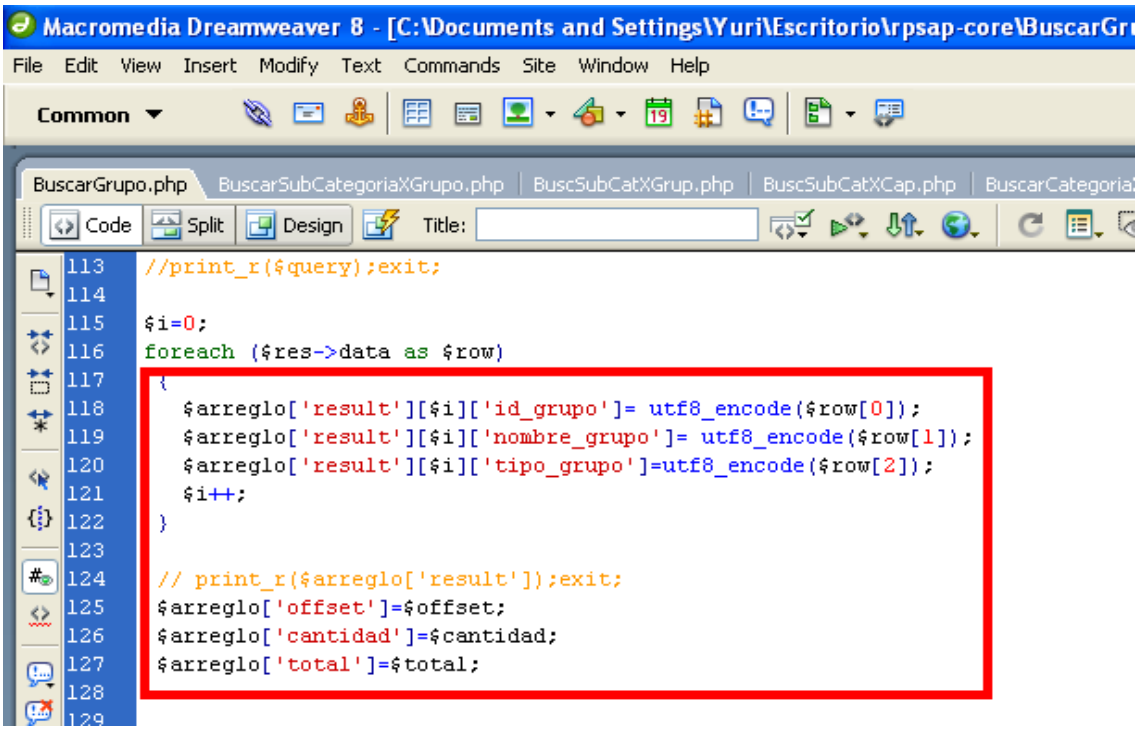


En el caso que se devuelva un arreglo que contenga otro arreglo dentro quedaría conformado de la siguiente manera:

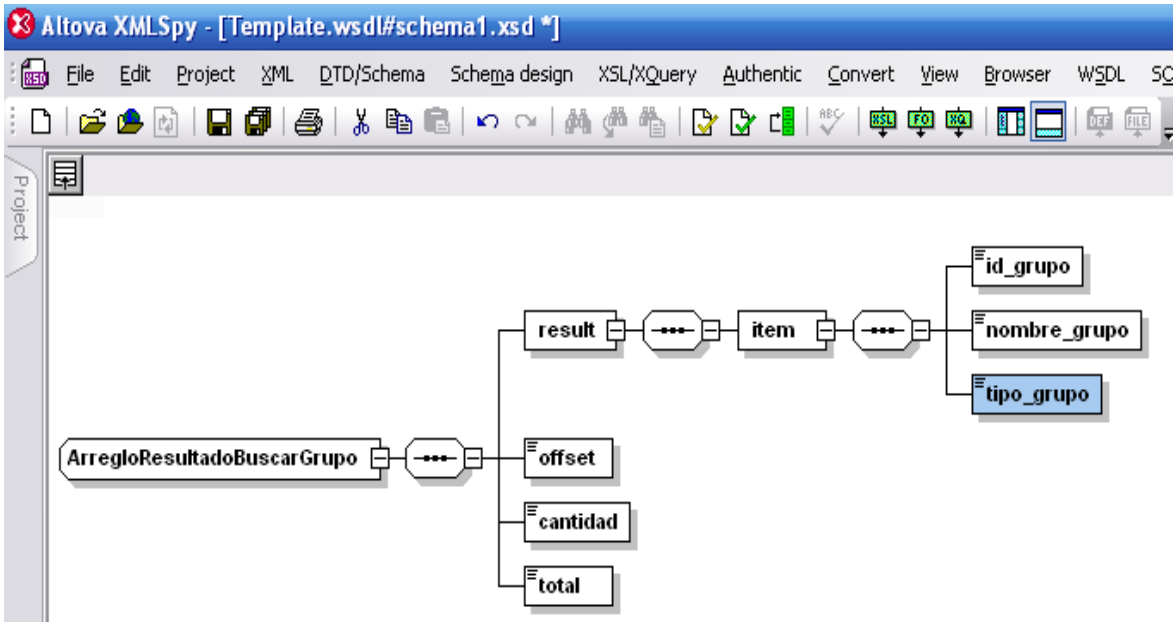

Altova XMLSpy - [Template.wsdl#schema1.xsd *]
File
Edit
Project
XML
DTD/Schema
Schema design
XSL/XQuery
Authentic
Convert
Project
ArregloResultadoBuscarGrupo
Estrategia
nombre
edad
offset
cantidad

Ejemplo de cómo quedarían los parámetros tomados desde la Pág. PHP:

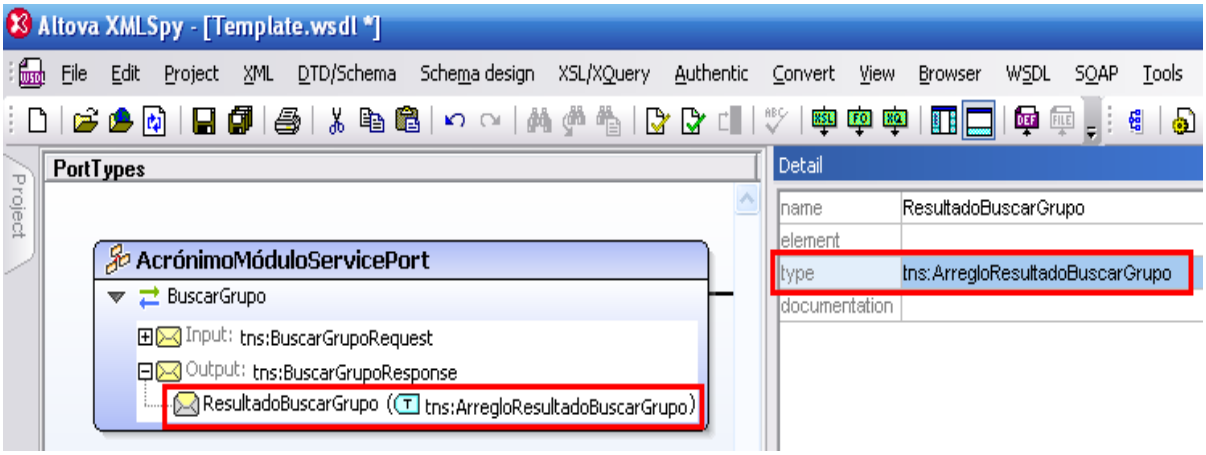
Tomando los parámetros que devuelve el método (Pág. PHP):



Quedando así el esquema del mismo:

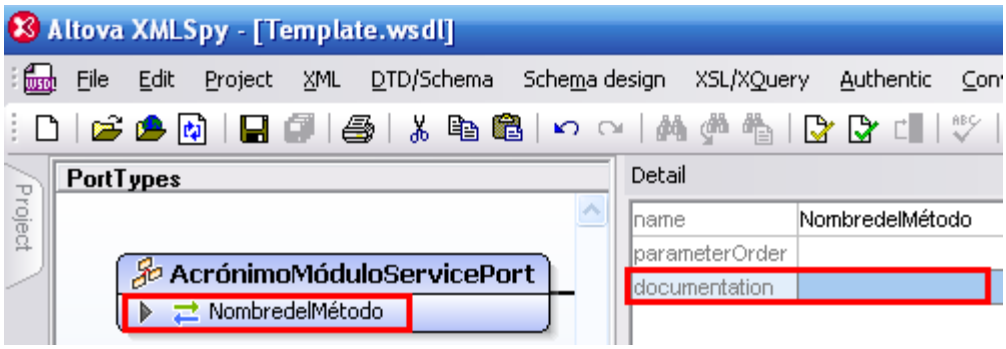


Ya una vez hecho el esquema entonces se le debe decir a la operación (en el **portType**) cual es el parámetro de salida (Output), para ello ya se debe haber salvado el esquema para que salga en la lista de tipos de variables, y así seleccionarlo:

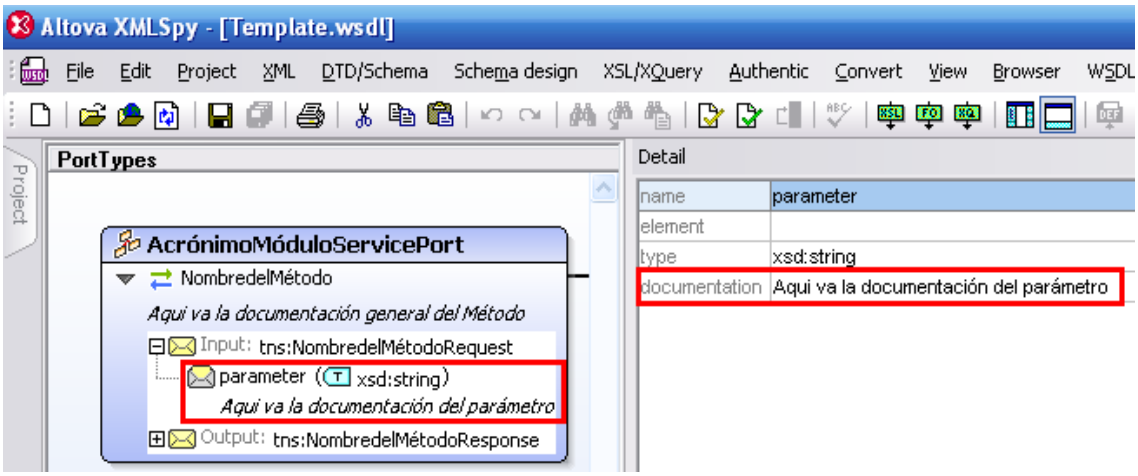


3.5.2.4-Agregando la documentación de las operaciones

Para agregar la documentación respectiva a cada operación, se da clic izquierdo en la operación y en *Detail,* se pone, en la parte *documentation,* las especificaciones de lo que hace la operación como tal, parámetros que se le pasan, la respuesta que devuelve, etc.



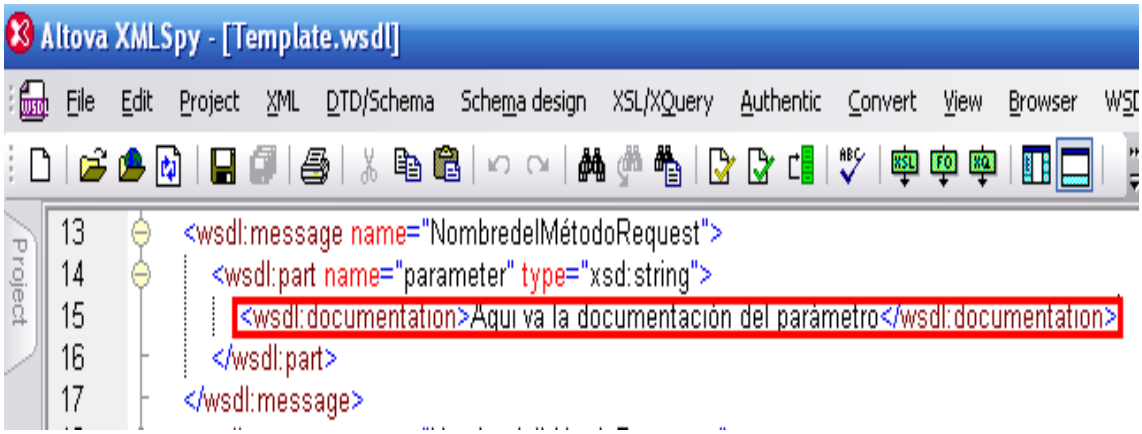
En caso de que se le agregue documentación a los parámetros (especificando lo que indica el mismo), quedaría de la siguiente forma:



Cada vez que se agrega una documentación, en el xml se ve como el elemento wsdl:documentation esta presente como el primer elemento hijo del elemento wsdl:part, tal y como se define en la especificación:

2.9.3.11- WSDL y el elemento *documentation*.

Planteada dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.



Aclarar que cada vez que se añade una nueva operación, esta se suma también a los métodos u operaciones del **bindings**.

3.5.3- Especificaciones a tener en cuenta en el elemento *bindings*

Solo se creará un Binding que se llamará "Acrónimo del Módulo+ServiceBinding" y se enlazará con el puerto creado, además el tipo del binding debe ser "soap", el estilo "rpc" y el protocolo de transporte sea "http://schemas.xmlsoap.org/soap/http".

El protocolo de transporte esta definido según la especificación:

2.9.8.2- Transporte HTTP

Planteada dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

Ejemplo código:

```
<binding name="RLServiceBinding" type="tns:RLServicePort">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
```

Se crea un soapAction por cada operación definida en el puerto. Las operaciones deben tener el nombre idéntico a los métodos php a los que se enlazarán y en el soapAction de estas se pondrá el acrónimo del módulo con el cual se está trabajando, un punto y el nombre de método php. Por último se deben especificar los atributos del input y output de cada operación.

Antiguamente para el atributo body del input se definía que el uso era "encoded" y el estilo de codificación http://schemas.xmlsoap.org/soap/encoding/", pero el PB elimina el uso de la codificación y especifica que estos atributos, tanto el body del input, como el body del output deben ser “literal”, lo anterior está descrito en las especificaciones:

2.9.8.3- Consistencia del atributo style.

2.9.8.4- Codificación y el atributo use.

Planteadas dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

También se le agregará la siguiente cabecera soap al input:

```
<soap:header message="tns:Header" part="Token" use="literal"/>
```

Para el atributo body del output se debe definir que el uso es "literal" como se planteó anteriormente**.**

Ejemplo código:

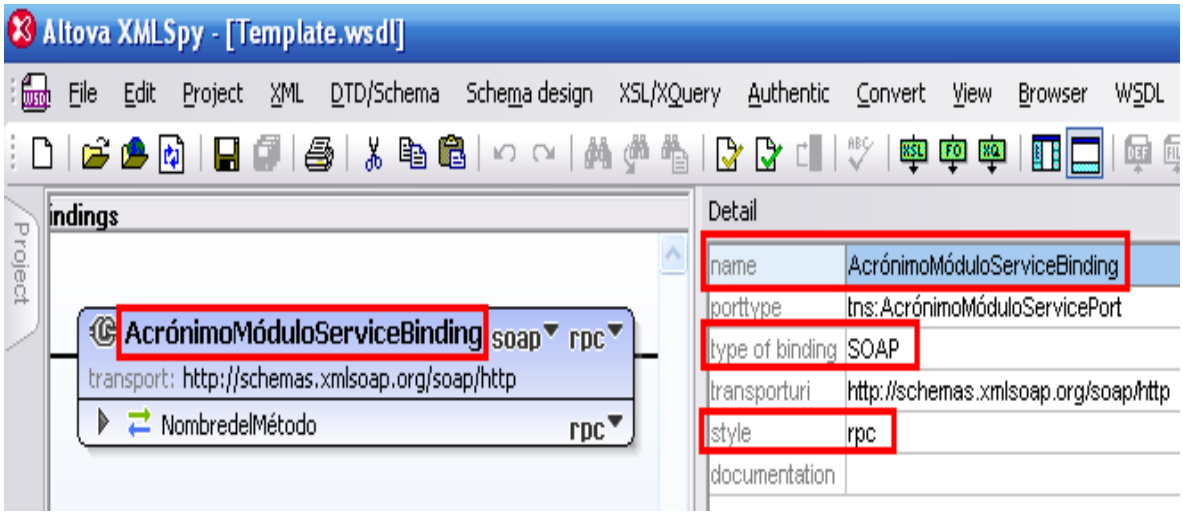
```
<wsdl:operation name="BuscarTuplaZona">
  <soap:operation soapAction="RL.BuscarTuplaZona"/>
    <wsdl:input>
    <soap:body use="literal"
    <soap:header message="tns:Header" part="Token" use="literal"/>
    </wsdl:input>
    <wsdl:output>
    <soap:body use="literal"/>
   </wsdl:output>
</wsdl:operation>
```

Se debe tener presente que el número de operaciones del **bindings** debe ser igual al número de operaciones del portType, al cual hace referencia, expuesto en la especificación:

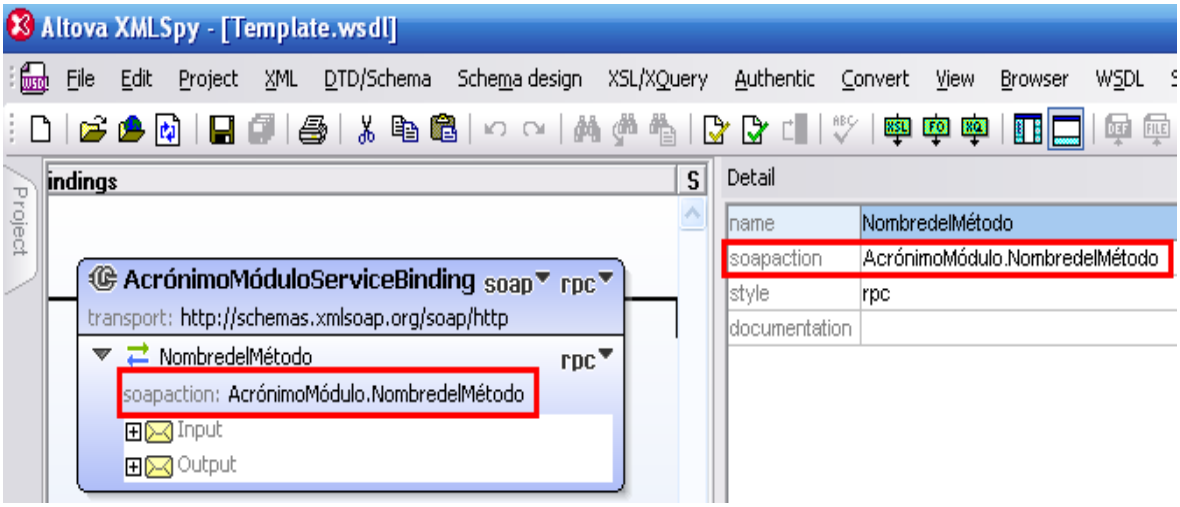
2.9.8.11- Consistencia de los elementos portType y bindings.

Planteada dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

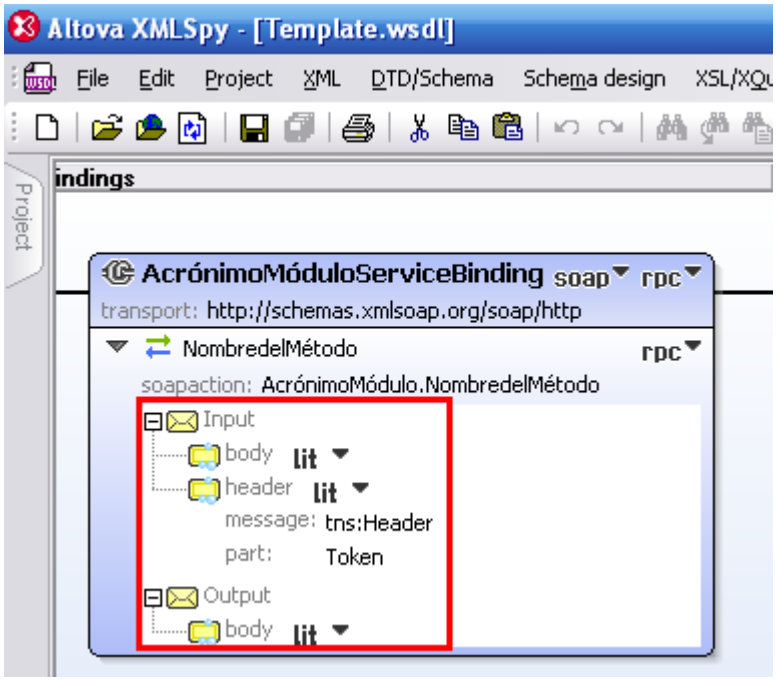
En este aspecto se debe tener presente como se pone el nombre, el tipo de bindings, y el estilo, los cuales quedan bien claro reflejados en la siguiente figura:



Ya en el soapaction se pondría el nombre del módulo con el cual se esta trabajando, un punto, y el nombre de método (AcrónimoMódulo.NombredelMétodo):



Ahora se muestra como quedarían los body del input y output con el atributo *use* en “literal” y el uso de la cabecera “Header” en el input:



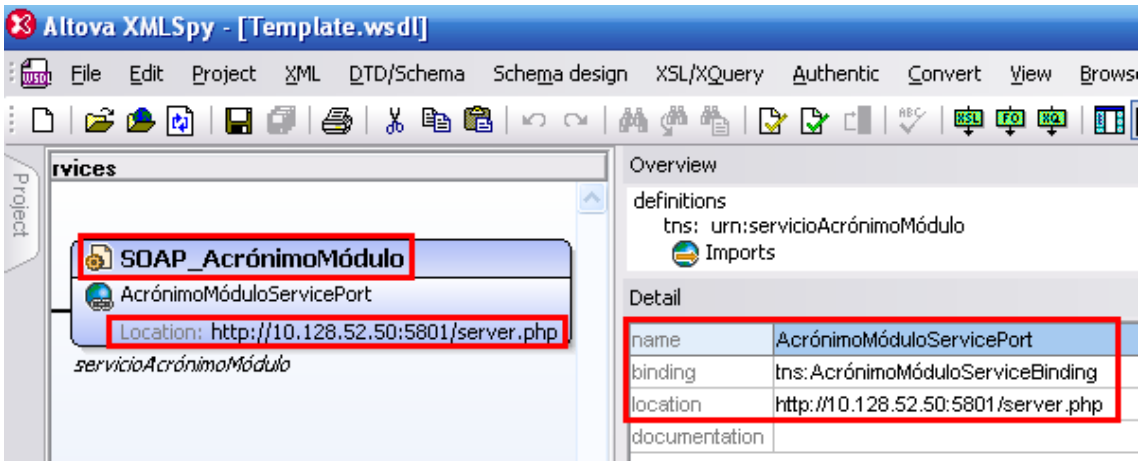
3.5.4- Especificaciones a tener en cuenta en el elemento *service*

Se debe crear un solo servicio que se debe nombrar como “SOAP_ + Acrónimo del módulo”. El nombre del puerto del servicio debe ser “Acrónimo del módulo + ServicePort” y el binding debe coincidir con el nombre definido para este, pero con el prefijo “tns:”.

Por último en el cuerpo del puerto se debe definir la dirección (address) del servidor contra el cual se va a probar la descripción (wsdl), que debe ser un url válido y se pondrá en el atributo localización (location) del address.

```
<service name="SOAP_RL">
<port name="RLServicePort" binding="tns:RLServiceBinding">
<soap:address location="http://10.128.52.50:5801/server.php"/>
</port>
</service>
```

En la figura que se muestra a continuación queda definido bien claro como poner los distintos nombres, y el atributo localización del Service:



Al definir un solo puerto (port) se evita una posible confusión que pudiera existir, al declarar dos puertos con destinos finales iguales por ejemplo, por tal motivo se debe tener presente la especificación:

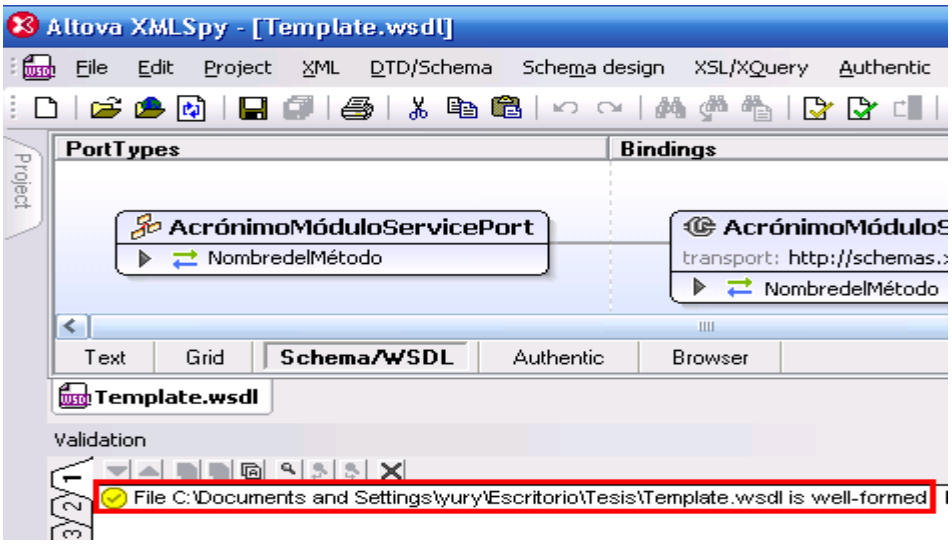
3.3.2.8.7- Múltiples puertos en un punto final.

Planteada dentro del epígrafe **2.9- Definiciones y especificaciones planteadas por el Perfil Básico** del capitulo II.

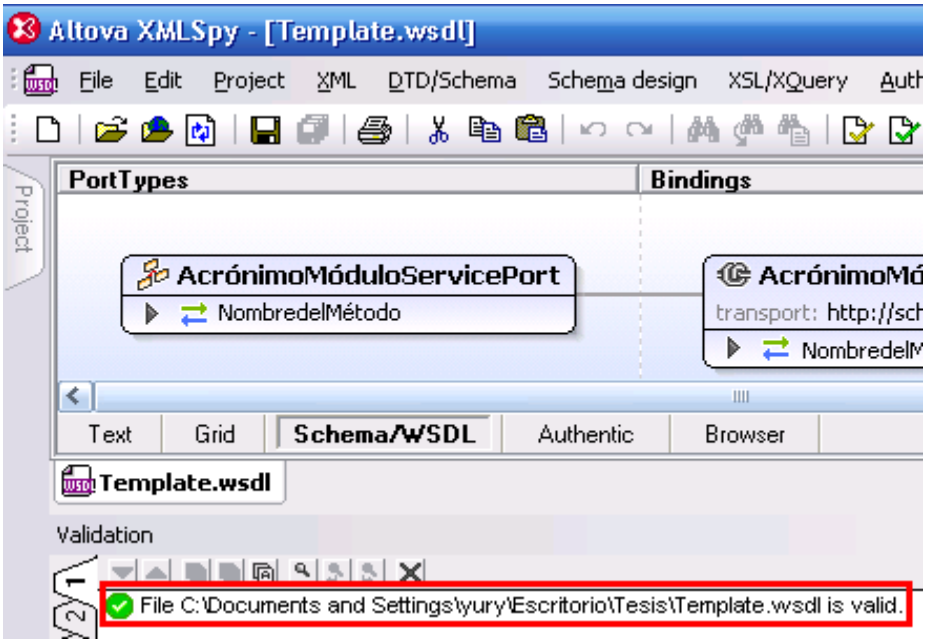
3.5.5- Verificación de la validez y la correcta estructura de la descripción

Ya una vez hecho todo lo descrito anteriormente en el documento, entonces se verificaría que el WSDL este bien formado y que sea válido, para ello se pulsaría **F7** y luego que se verifique que está bien formado se pulsaría **F8**, para verificar si es válido.

Ejemplo de WSDL bien formado:

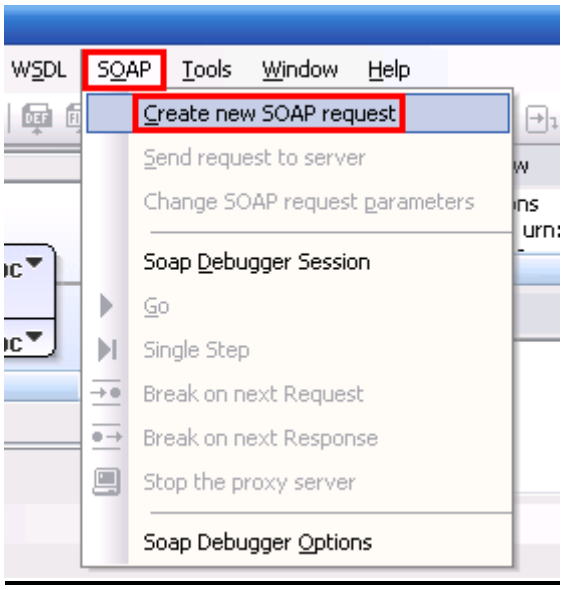


Ejemplo de WSDL válido:

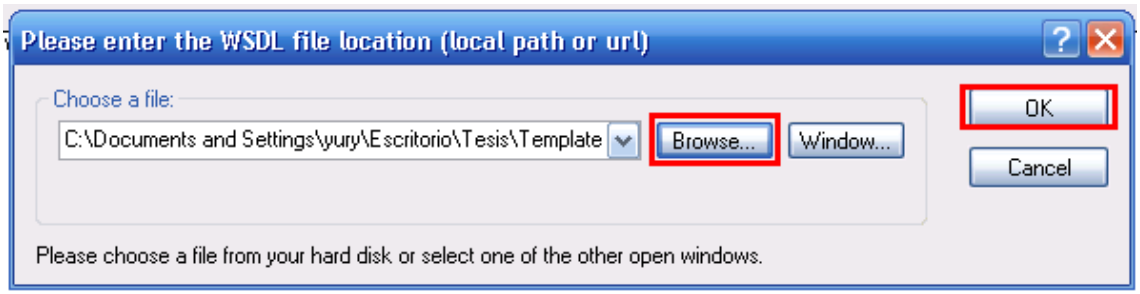


3.5.6- Realización de pruebas contra el servidor

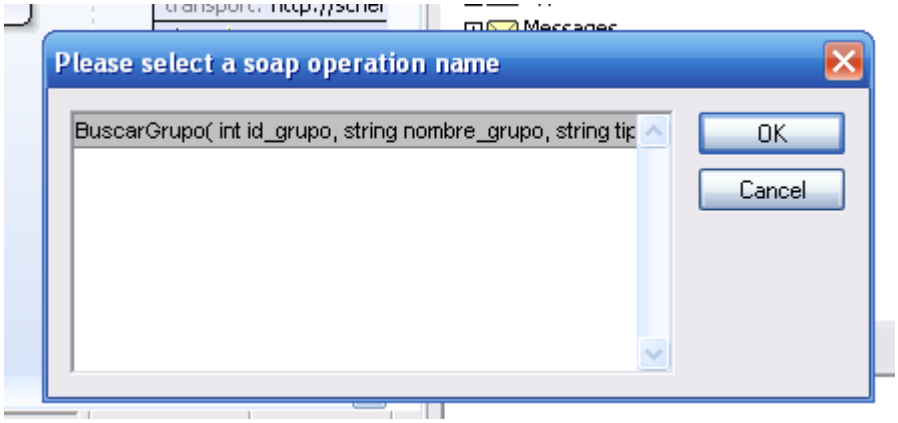
Se debe ir a la barra de menú, dar clic izquierdo en el menú SOAP y crear una nueva consulta:



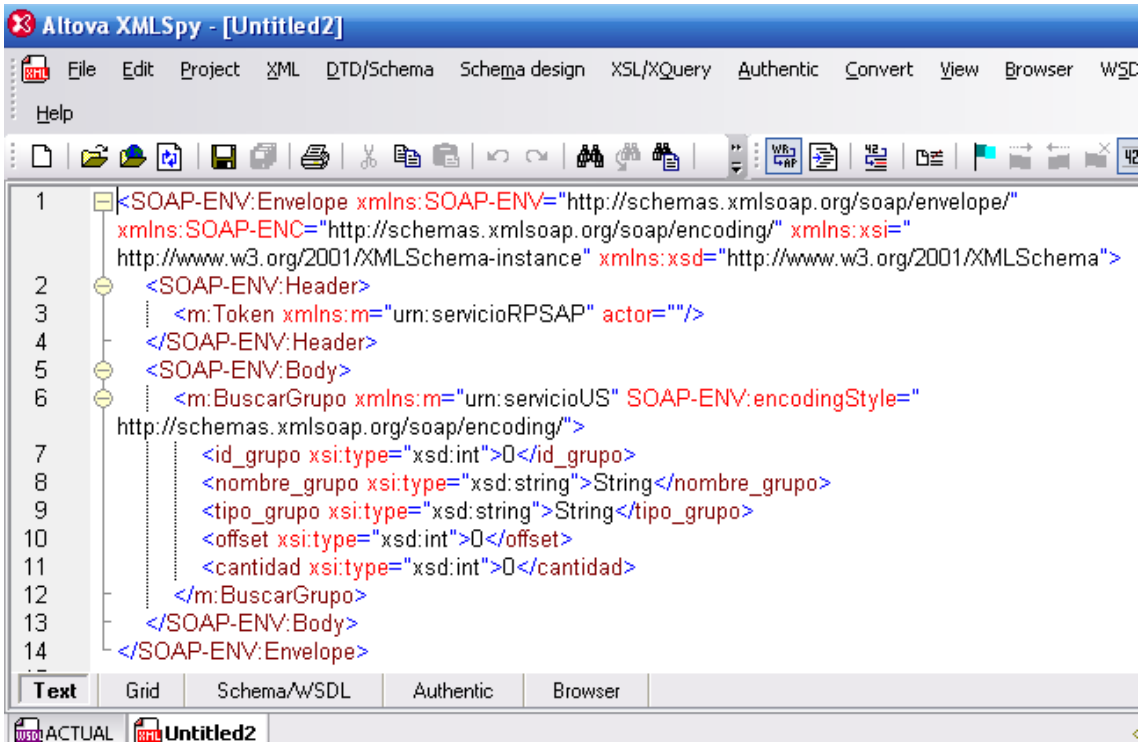
Luego se selecciona la ubicación del wsdl al cual se le haría la consulta (Se da clic en *browse* y se busca donde esté guardado, luego se da clic en OK):



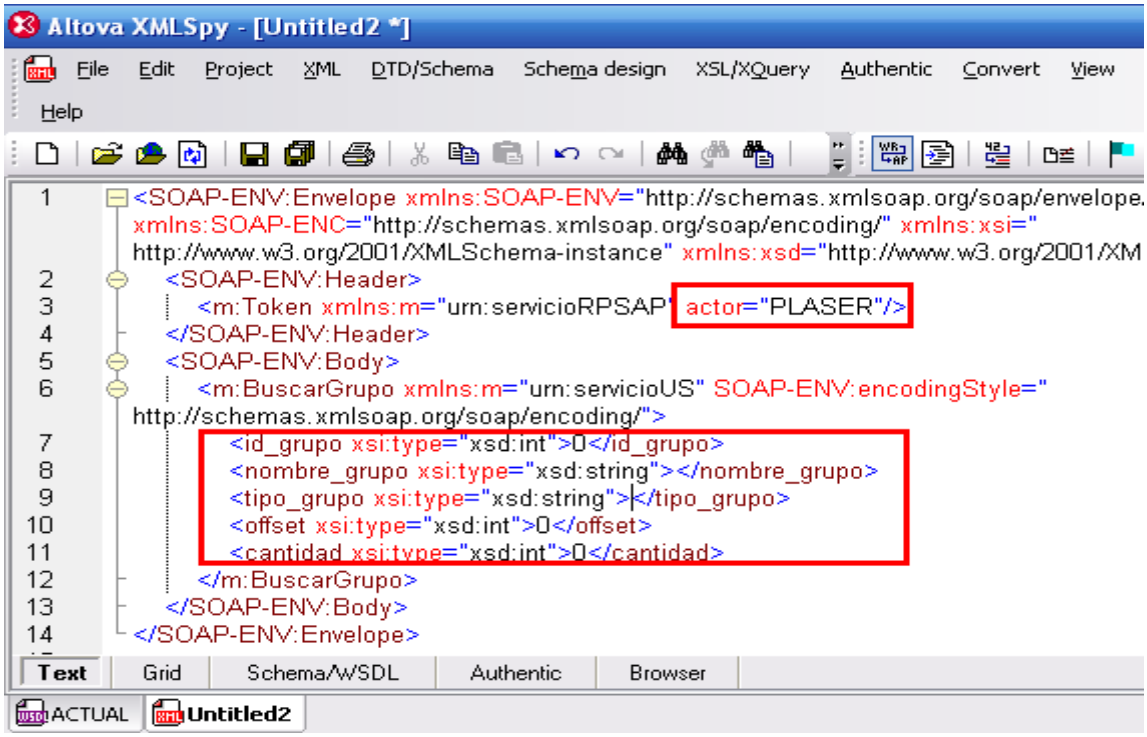
Posteriormente aparecería una ventana que contiene todos los métodos que están presentes en el wsdl con el cual se esta trabajando, se escoge el método al cual se le haría la consulta y se da OK:



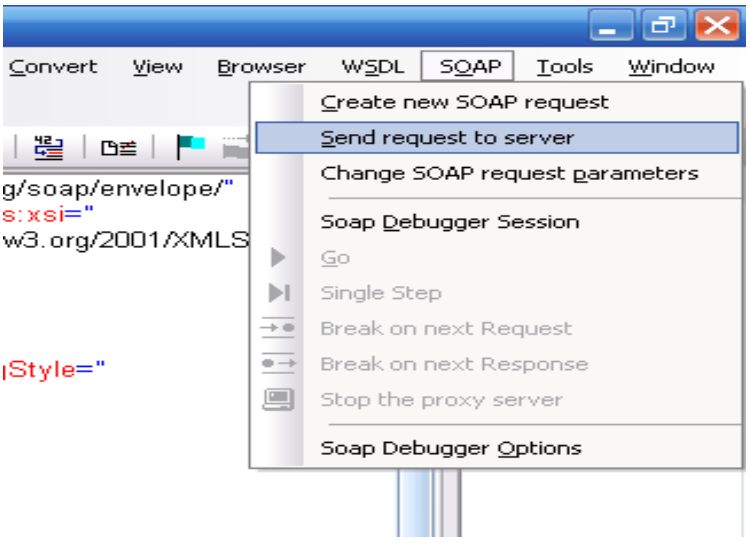
Aparecería un XML:



Pero al mismo se le debe cambiar el actor (Siempre es PLASER), así como poner en blanco los parámetros string y los int en cero:

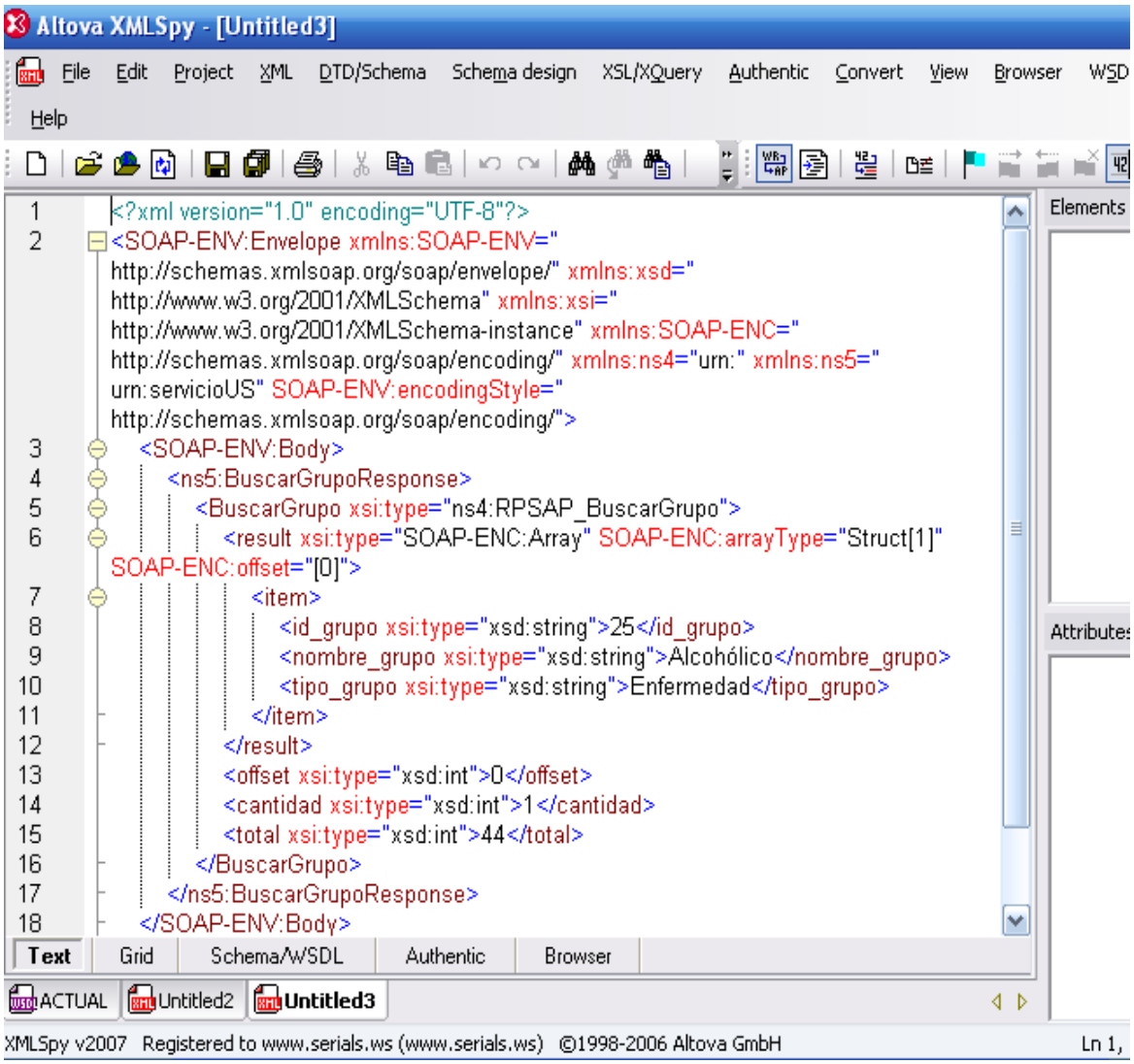


Una vez chequeado esto se iría nuevamente al menú SOAP y se enviaría la consulta al servidor

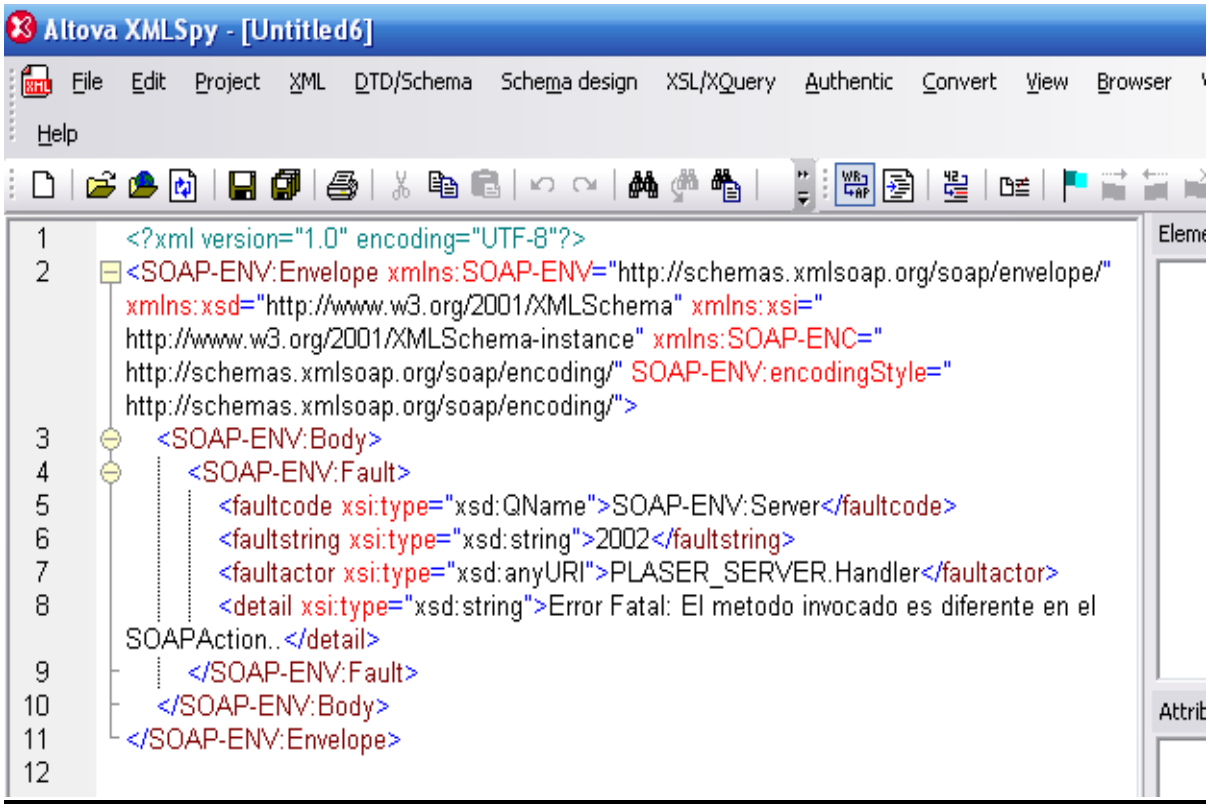


Realizado el paso anterior, entonces saldría otro XML en el cual se vería el resultado de la consulta como tal, en caso de error saldría un XML, con el error donde se contiene

Sin error:



Con error:



Los pasos de verificar que el WSDL esté bien **formado** que **sea válido** y **probarlo contra el servidor** es aconsejable que se vaya chequeando cada vez que se termine un método puesto que seria más fácil de detectar donde esta el error y proceder a su corrección.

3.5.7- Clientes en PHP 5

Una vez que se haya realizado la descripción de manera correcta, se procede a la confección del cliente en PHP, para una mayor facilidad, se muestra a continuación de manera íntegra, el código de los mismos; tener en cuenta las situaciones particulares del wsdl creado y entonces adaptar el código del cliente en PHP, a las necesidades encontradas.

Es importante aclarar que antes de encuestar el servicio, mediante el cliente PHP, el usuario se debe haber autenticado previamente en el componente de seguridad del proyecto SAAA .

```
<?php
 /*
Este código sirve como cliente para el Módulo + NombredelMódulo.
*/
// Se inicia la sesión
session_start();

/*
La clase a continuación se usa para ver el mensaje SOAP que confecciona PHP5, antes de realizar la petición.
*/
class ClienteSoap extends SoapClient
{
  function __call($funcion,$args)
  {
   return parent::__Call($funcion,$args);
  }
  public function __doRequest($request, $location, $action, $version)
  {
  header("Content-Type: text/xml");
  echo $request;exit;
  }
}

/*
Aquí se instancia la clase que sirve para invocar los Métodos del Servicio Web, notar que:
1- El 1er parámetro indica la localización del wsdl, el resto se mantiene igual.
Ahora si en vez de instanciar la clase nativa (SoapClient), que se muestra a continuación, se instancia la clase anterior (ClientSoap), los parámetros del constructor se mantienen, se ve el mensaje SOAP que se debe enviar y no la respuesta en sí, lo mismo sirve para comparar el mensaje SOAP que genera la herramienta Altova XMLSpy, con la respuesta de esta clase.
*/
$client = new SoapClient("NombreWsdl.wsdl",array("style"=> SOAP_RPC,"use"=> SOAP_ENCODED));

/* Se construyen los parámetros de entrada(Lo mismo es acorde a cada método u operación descrita en el wsdl), los valores asignados, no son obligatorios, los mismos son a modo de ejemplo.
*/
$in_us = array();
$in_us['arregloid'][0]=37;
```

```
$in_us['arregloidprov'][0]=3;
$in_us['offset']=0;
$in_us['cantidad']=10;
$in_us['tipo_salida']=0;

// Finaliza  la estructura del arreglo

// Se confecciona el Token de Seguridad (Es igual siempre)
$header[]= new SoapHeader('plaser.softel.cu', 'Token', $_SESSION['token'], true,PLASER);

try
{
/*Se invoca el método correspondiente en el wsdl, notar que el Soapaction en igual al definido en el Binding, para la operación, dentro del wsdl.
*/
$retval=$client->__soapCall("NombreOperación",
$in_us, array('soapaction'=>'AcrónimoMódulo.NombreOperación'),$header);


print_r($retval);
}
catch (SoapFault  $E)
{
print $E->detail .'<br>'.$E->faulcode.'<br>'.$E->faulstring.'<br>'.$E->faulactor;
}

?>
```

La confección de este cliente (uno por cada operación definida en el wsdl) se realiza con el objetivo de comprobar, que el resultado xml devuelto por el servicio coincida con la estructura de dato complejo definido para la salida de la operación, ya que la herramienta Altova XMLSpy, a la hora de generar la respuesta, muestra el xml acorde al resultado que devuelve el método del negocio, al cual se está encuestando. Pero si al verificar con el cliente, la información no se muestra de la forma esperada, entonces se debe revisar el esquema del dato complejo definida para la salida de la operación asociada al servicio encuestado, pues a la hora de confeccionarlo se han de haber introducido errores.

Conclusiones

Como conclusión al presente capitulo, se puntualiza que en el mismo se dieron a conocer los pasos necesarios para realizar el procedimiento que mejorará la interoperabilidad de los servicios web. Todo esto fue posible teniendo en cuenta todo lo abordado a lo largo de investigación y en particular teniendo en cuenta las restricciones/especificaciones del perfil básico de la WS-I.

CONCLUSIONES

La aplicación de este procedimiento en la empresa Softel, específicamente en el proyecto APS ayuda a obtener un proceso de elaboración de las descripciones de los servicios Web (wsdl) más robusta, organizada y estandarizada respecto a las normas actuales definidas por la WS-I, la organización que promueve la interoperabilidad de los servicios Web, permitiendo la comunicación entre distintos sistemas y lenguajes de programación, por ello:

1. Fue necesario la realización de un estudio de los perfiles de la WS-I para lograr en los servicios la conformidad con respecto a la comunicación que se requería.

2. Se valoró los servicios web de la empresa Softel, haciendo un énfasis más exhaustivo con los desarrollados por el proyecto APS, para ver la situación en la que se encontraban con respectos a los perfiles regidos por la organización que se dedica a la interoperabilidad.

3. Se hizo un estudio minucioso y amplio acerca de los perfiles definidos por la WS-I, con el objetivo de entenderlos y así poder trabajar con ellos de forma eficiente y correcta.

Todo esto trajo consigo la propuesta de procedimiento que se muestra; conclusión de un trabajo investigativo para mejorar la interoperabilidad, el desarrollo y evolución de los distintos servicios web.

RECOMENDACIONES

Los autores recomiendan:

- Aplicar el procedimiento propuesto a los distintos proyectos de la universidad y en específico a los de la facultad 7.
- Continuar la investigación sobre el tema, teniendo en cuenta las nuevas versiones o mejoras que puedan surgir respecto a los perfiles, que se definan por la WS-I, relacionados con la interoperabilidad y que aporten nuevas mejoras, que puedan ser aplicadas a los servicios web.

REFERENCIAS BIBLIOGRÁFICAS

1. BARRERA, Y. P. R. O. P. Buenas Prácticas WSDLs, 2007: 7.
2. DESARROLLOWEB.COM. *WSDL para la documentación de Servicios Web*. Disponible en: http://www.desarrolloweb.com/articulos/1581.
3. W3C. *Web Services Description Language (WSDL) 1.1*, 2001. Disponible en: http://www.w3.org/TR/2001/NOTE-wsdl-20010315.
4. DESARROLLOWEB.COM. *Servicios Web en plataforma .NET*. Disponible en: http://www.desarrolloweb.com/manuales/54/.
5. THESERVERSIDE. *Introducción a los Servicios Web en Java*. Disponible en: http://www.programacion.net/java/tutorial/servic_web/.
6. NOCIONDIGITAL.COM. *Servicios Web con PHP (NuSOAP)*. Disponible en: http://www.nociondigital.com/webmasters/php-tutorial-servicios-web-con-php-nusoap-detalle-168.html
7. Ídem referencia 4.
8. Ídem referencia 4.
9. SKONNARD, A. *Mejorar la interoperabilidad de los servicios Web*, 2006. Disponible en: http://www.microsoft.com/spain/interop/developers/improvingWSInterop.mspx
10. WS-I. *Basic Profile Version 2.0*, 2007. Disponible en: http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html
11. Ídem referencia 10.
12. Ídem referencia 10.
13. W3C. *XML Schema Part 1: Structures Second Edition*, 2004. Disponible en: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/
14. W3C. *XML Schema Part 2: Datatypes Second Edition*, 2004. Disponible en: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

BIBLIOGRAFÍA

1. *Arquitectura Orientada a Servicios (SOA)*. 1. *WSDL*. Disponible en: http://www.cibernetia.com/manuales/servicios_web/4_wsdl.php
2. *Protocolo simple de acceso a datos (SOAP)*. Disponible en: http://www.monografias.com/trabajos29/protocolo-acceso/protocolo-acceso.shtml
3. BARRERA, Y. P. R. O. P. Buenas Prácticas WSDLs, 2007: 7.
4. BELL, E. A. M. M. *Service-Oriented Architecture (SOA)*. 2006. 388 p. 9780471768944
5. CAUDWELL, P. *Servicios Web XML: Profesional*. 798 p. 9788441513631
6. CERAMI, E. *Web Services Essentials*. 289 p. 9780596002244
7. DESARROLLOWEB.COM. *Servicios Web en plataforma .NET*. Disponible en: http://www.desarrolloweb.com/manuales/54/
8. DESARROLLOWEB.COM. *WSDL para la documentación de Servicios Web*. Disponible en: http://www.desarrolloweb.com/articulos/1581.php
9. ERL, T. *Service-Oriented Architecture (SOA)*. 2005. 804 p. 9780131858589
10. ERL, T. *SOA Principles of Service Design*. 2008. 612 p. 9780132344821
11. GONZÁLEZ, A. G. R. R. M. *XML a través de ejemplos*. 500 p. 8478974555
12. IVERSON, W. *Real World Web Services*. 2005. 224 p. 9780596006426
13. KULCHENKO, J. S. D. T. P. *Programming Web Services with SOAP*. 2002. 268 p. 9780596000950
14. LEQUERICA, J. R. *Web Services*. 336 p. 8441515379
15. MARGOLIS, B. *SOA for the Business Developer*. 2007. 330 p. 9781583470657
16. MEHTA, N. A. T. *UDDI*. 415 p.
17. MONSON-HAEFEL, R. *J2EE Web Services*. 120 p. 9780321146182
18. NEWCOMER, E. *Understanding Web Services*. 325 p. 9780201750812
19. NOCIONDIGITAL.COM. *Servicios Web con PHP (NuSOAP)*. Disponible en: http://www.nociondigital.com/webmasters/php-tutorial-servicios-web-con-php-nusoap-detalle-168.html
20. OMMEREN, M. V. D. B. N. B. E. V. *SOA for Profit*. 2007. 262 p. 9789075414141

21. PEUSER, O. Z. M. T. S. *Perspectives on Web Services*. 2003. 682 p. 9783540009146

22. RAFTER, D. H. H. *Beginning XML*. 2007. 1258 p. 9780470114872

23. RAY, E. *Learning XML*. 2003. 418 p. 9780596004200

24. SHORT, J. S. A. S. *Creación de Servicios Web XML para la plataforma .NET*. 500 p. 8448137027

25. SKONNARD, A. *Mejorar la interoperabilidad de los servicios Web*, 2006. [Disponible en: http://www.microsoft.com/spain/interop/developers/improvingWSInterop.mspx

26. THESERVERSIDE. *Introducción a los Servicios Web en Java*. Disponible en: http://www.programacion.net/java/tutorial/servic_web/

27. W3C. *Web Services Description Language (WSDL) 1.1*, 2001. Disponible en: http://www.w3.org/TR/2001/NOTE-wsdl-20010315

28. W3C. *XML Schema Part 1: Structures Second Edition*, 2004. Disponible en: http://www.w3.org/TR/2004/REC-xmlschema-1-20041028/

29. W3C. *XML Schema Part 2: Datatypes Second Edition*, 2004. Disponible en: http://www.w3.org/TR/2004/REC-xmlschema-2-20041028/

30. WS-I. *Basic Profile Version 2.0*, 2007. Disponible en: http://www.ws-i.org/Profiles/BasicProfile-2_0(WGD).html

31. ZAMORA, A. R. *Publicación en Internet y Tecnología XML*. 464 p. 8478975853