UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

CRITERIOS DE SELECCIÓN PARA LAS HERRAMIENTAS DE ORQUESTACIÓN DE SERVICIOS WEB

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

JAIME CRISTIAN ACEVEDO EMALDIA

PROFESOR GUÍA:
MARIA CECILIA BASTARRICA PIÑEYRO

MIEMBROS DE LA COMISIÓN:
CARLOS HURTADO LARRAIN
SANDRA XIMENA DE LA FUENTE GONZALEZ

SANTIAGO DE CHILE
ENERO 2007

Resumen

Cuando fue creado el Web, quizás nunca se pensó el alcance que iba a llegar a tener, probablemente sólo se pensó en hacer algo simple y que permitiera compartir la información rápidamente. Ahora bien el hecho de compartir información es la base del mundo de hoy y en general podemos decir que el mundo se mueve gracias a que se comparte información y dicha información se ocupan en los distintos procesos de negocios que poseen las empresas.

Si bien el Web es claramente un medio para compartir información, con el tiempo se ha ido buscando que sea capaz de entender, modelar o ejecutar los procesos de negocios para así hacer más fácil y rápido el trabajo de las personas. Por esto el Web ha ido cambiando su forma y adoptando nuevos y más protocolos como los SOAP, XML, WSDL, etcétera, que ha llevado al Web a formar parte de un SOA (*Service Oriented Architecture*). Las empresas quieren interactuar con sus procesos internos y con otras empresas por lo que se crea un nuevo concepto, el de Orquestación de Servicios Web (*WSO, Web Services Orchestration*) que permite expresar y ejecutar la lógica del negocio en el Web. Cuando se construye una solución orientada a servicios, la orquestación provee un significativo atractivo en el control y encubrimiento de la representación de la lógica del proceso a automatizar.

La WSO esta basada en reglas para los procesos de negocios que son ejecutadas por un Motor de Orquestación, el cual toma los datos de un *scripts* escritos en un lenguaje. Hoy en día hay dos grandes movimientos para dicho lenguaje BPEL4WS (Bussiness Process Execution Lenguaje for Web Services) y BPML (Bussines Process Manager Lenguage), siendo ambos muy parecidos. En general las herramientas de orquestación son capaces de trabajar con cualquiera de los dos, por lo que no genera mayores inconvenientes.

Cuando se quiere implementar este tipo de tecnología en la empresa nacen muchos cuestionamientos de que elementos son importantes considerar en cada herramienta a evaluar para poderlas comparar de mejor manera en el momento de decidir que herramienta implementar. Se presenta una tabla con los elementos a evaluar para facilitar el trabajo de la comparación de dichas herramientas.

La tabla consta de cuatro puntos principales como son: Caracteristicas de Programa, Procesos de Negocios (BPM), BPEL (Bussines Process Execution Lenguaje) y Motor ESB (Entrepise Service Bus). Dentro de cada uno de estos puntos se han considerado varios otros elementos para que la evaluación de las herramientas sea lo más precisa y simple, para poderlas comparar con la mayor cantidad de información expuesta de una manera ordenada y fácil de leer.

Se aplicó dicha tabla a tres herramientas lo que permite observar en forma clara el cumplimiento o no de los elementos evaluados en cada una de las herramientas.

Finalmente podemos decir que si bien orquestación apunta a un futuro muy prometedor, aún hace falta avanzar en la implementación de estándares y en buscar una solución real de compatibilidad para los elementos ya desarrollados, para no tener que volver a programar adaptándolos a las nuevas tecnologías.

AGRADECIMIENTOS

Agradezco a todos los ayudaron a ser posible el desarrollo de esta tesis, a mis padres que me apoyaron en todos los aspectos, a las personas que han estado a mi lado mientras la he desarrollado, así como todos aquellos que de alguna forma han estado atento a mis avances y logros en la misma.

Agradezco al proyecto FONDECYT N° 1050642 por financiar parcialmente esta memoria.

Índice

Anexos

Figuras

1. Introducción

Originalmente el Web fue creado para compartir información entre los científicos. Hoy en día la ocupan gobiernos, empresas e individuos para que su información sea accesible vía "Web". Sin embargo, gran cantidad de la información que hoy se comparte en el Web es entendible sólo para los humanos o aplicaciones hechas a la medida para entenderla. Por esto lo que se busca es transformar el Web en un medio a través del cual la información pueda ser compartida, entendida y procesada por herramientas de forma automática [8].

Uno de los conceptos que están siendo desarrollados para poder lograr transformar el Web es el de servicios Web. Este lo podemos definir como una serie de funcionalidades relacionadas que se pueden acceder en forma programada a través del Web [7]. Un servicio Web funciona encapsulando un rango de funciones que van desde una simple respuesta solicitada hasta un completo proceso de negocios [8]. Este poderoso concepto está siendo gradualmente tomado en cuenta debido a la convergencia de empresas y gobiernos para hacer del Web el lugar para realizar todo tipo de actividades. Uno de los más importantes asuntos al respecto es el uso del Web como facilitador de la externalización de servicios. Este nuevo modelo podría hacer factible para las compañías conseguir significativas reducciones de costos, rápido despliegue de soluciones de actividades y estar abiertas a nuevas oportunidades de negocios. Así podemos definir dos tipos de servicios: los simples (donde no es necesaria la intervención de otros servicios Web) y los compuestos (donde interviene más de un servicio Web) [7].

Los procesos de negocios hoy en día necesitan de agilidad y rapidez para adaptarse a las necesidades y condiciones del mercado. Esto puede incluir nuevos clientes, socios, o materiales para ellos. Un estándar único debe poder manejar las interacciones entre los servicios Web tanto de la Arquitectura de Integración de la Empresa (EIA) como de los B2B [1]. Orquestación de Servicios Web (WSO, sigla en inglés) se trata sobre proveer un estándar base para acercarse a la conectividad de los servicios Web mediante un lenguaje de alto nivel para procesos de negocios. Estándares como BPEL4WS, WSCI y BPML han sido diseñados para reducir la complejidad requerida para la orquestación de servicios Web, reduciendo así también el tiempo para la venta y el costo de desarrollo y mantenimiento, incrementan la eficiencia en conjunto y la exactitud en los procesos de negocios. Sin un estándar en común, cada organización debería desarrollar su propio conjunto de protocolos para los procesos de negocios, que además de los propios costos de desarrollo dejaría poca flexibilidad para la colaboración entre los distintos sistemas de orquestación para servicios Web tanto dentro de la propia empresa como entre las distintas empresas [2].

La orquestación de servicios Web es una forma particular de composición de servicios Web. Dicha composición se logra a través de un lenguaje de composición o algún tipo de código que permite invocar una serie de operaciones relacionadas con la composición de los servicios Web. Por esto un WSO no necesariamente necesita una interfaz de servicio Web, sino que necesita un motor de WSO para poder ser ejecutado [4].

Los primeros trabajos sobre WSO fueron desarrollados con lenguajes como eCo, WSCL, XLANG y WSFL. Estos lenguajes fueron creados para mostrar cada uno de ellos una parte importante de WSO. Por ejemplo, eCo fue creado para mostrar el valor de integrar los servicios e-commerce, con un foco en el intercambio de los documentos para la integración entre los B2B. La especificación tuvo una vaga noción de orquestación, mostrando cómo un proceso podía estar

compuesto de un servicio Web. En la actualidad se han reunido IBM, Microsoft y BEA para crear un lenguaje llamado BPEL4WS (Business Process Execution Lenguage for Web Services). Este es una especificación del comportamiento del servicio Web en un proceso de negocio. La especificación brinda una gramática basada en XML para describir la lógica del control requerido para coordinar los servicios Web que participan en el proceso. Asimismo, Sun en conjunto con otras empresas como Intalio, CSC y otros, se reunieron y formaron dos lenguajes que en conjunto los podemos comparar a BPEL4WS; ellos son: WSCI (Web Service Choreography Interface) y BPML (Business Process Management Language). Estos dos lenguajes en conjunto tienen un tratamiento y comportamiento muy similar al de BPEL4WS. BPML también provee una gramática en XML que describe y maneja la lógica del proceso. Esta gramática puede ser interpretada y ejecutada por un motor de orquestación, el cual es controlado por uno de los participantes [5].

El Motor de Orquestación está encargado de coordinar las variadas actividades en el proceso, y compensar el sistema cuando ocurre algún error. El motor de orquestación en general está incluido en las herramientas para diseño y manejo de los procesos de negocios. Algunas de estas herramientas son BEA WebLogic Workshop, Collaxa Orchestration Server, IBM's BPWS4J y SunONE WSCI Generator.

Dichas herramientas no están basadas en ningún estándar por lo que se desea poder definir un marco conceptual que permita compararlas que poseen o pretenden ser sistemas de orquestación de servicio Web. Esto se realizará definiendo en primera instancia qué es orquestación de servicios Web, luego en base a dicha definición se establecerán los elementos que debe poseer como mínimo y que puede llegar a tener un sistema de orquestación. Dentro de dichos elementos hay cosas como compatibilidad de lenguajes, conectividad y otras.

Una vez establecidos los elementos mínimos, se procederá a buscar un método para comparar estos elementos entre las distintas herramientas, como por ejemplo que tal o cual herramienta, es compatible con tal o cual lenguaje. En pocas palabras se espera poder realizar un cruce entre las herramientas que implementan sistemas de orquestación y los elementos mínimos que sean considerados generando una tabla de evaluación de las herramientas disponibles en el mercado para la orquestación de servicios Web.

Así mismo se espera poder generar dicha tabla para poder ser utilizada y extendida para futuras herramientas de orquestación.

1.1. Justificación

Cada vez son más las empresas que toman la decisión de desarrollar sus sistemas en plataformas basadas en servicios Web y cada vez es más necesario poder hacer que las distintas empresas coordinen sus servicios Web para que trabajen en forma conjunta o que al menos puedan ocupar todos los servicios que ofrecen los demás servicios Web. Para esto se ha propuesto una serie de lenguajes de orquestación para servicios Web. Para que dichos lenguajes sean funcionales, es necesario contar con un motor de orquestación para servicios Web. Este motor es un software desarrollado para entender, aplicar y ejecutar el lenguaje de orquestación. Básicamente el motor de orquestación interpreta los *scripts* que describen los procesos de negocios que a la empresa le interesa ejecutar.

Para poder diseñar los *scripts*, hoy en día existen herramientas de modelamiento que permiten que dicho proceso sea simple y rápido. Se puede así ser lo más ágil posible con respecto a los cambios que se presentan en el mercado para las empresas que tienen su potencial en los servicios vía Web.

Por esto es importante saber qué herramienta de modelamiento y qué motor de orquestación son más convenientes y apropiados, ya que no todos ofrecen las mismas posibilidades, lenguajes e interacciones con servicios Web.

Así es como tenemos varios software de distintos proveedores que nos permiten realizar orquestación de servicios Web como son BEA Weblogic Workshop, Microsoft BizTalk Server, Collaxa Orchestration Server, IBM BPWS4J, Intalio | n^3 entre otros.

1.2. Objetivos

1.2.1. Objetivo General

Establecer criterios objetivos para la selección de herramientas de orquestación de servicios Web.

1.2.2. Objetivos Específicos

- Estudiar el concepto de orquestación. Establecer cuáles son las características necesarias y deseables de un sistema de orquestación.
- Buscar, estudiar y analizar las distintas herramientas de orquestación disponibles en el mercado.
- Definir los criterios y métodos de evaluación de dichos criterios para la comparación de las herramientas de orquestación.
- Establecer una tabla con la cual serán evaluadas las distintas herramientas.
- Aplicar dicha tabla en las herramientas encontradas.
- Sacar conclusiones con respecto a las herramientas evaluadas.

2. Servicios Web

2.1. Definición de Servicio Web

Una definición genérica de Servicio Web es: "Una aplicación accesible a otras aplicaciones a través del Web". El UDDI consortium (www.uddi.org) introduce la siguiente definición "Aplicación de negocios modular autocontenida que está abierta, orientada a Internet y con una interfaz basada en estándares" (*Self-contained, modular business applications that have open, Internet-oriented, standards-based interfaces*). La referencia mundial en el mundo Web (W3C) lo define como "*un sistema de software identificado por una URI, cuyas interfaces públicas y enlaces se definen y describen usando XML. Su definición puede ser descubierta por otros sistemas de software. Estos sistemas pueden interactuar con el servicio Web de la forma prescrita por su definición, usando mensajes basados en XML a través de protocolos estándares de Internet*". Y por último, una definición precisa disponible en Webopedia (http://www.pcWebopedia.com/) dice "una manera estandarizada de integrar usos basados en el Web usando XML, SOAP, WSDL y UDDI abierta a estándares usando como espina dorsal el Protocolo de Internet, XML se utiliza para marcar los datos con etiqueta, SOAP se utiliza para transferir la data, WSDL se utiliza para describir los servicios disponibles, y UDDI se utiliza para el listado de qué servicios están disponibles" (*a standardized way of integrating Web based applications using XML, SOAP, WSDL and UDDI open standards over an Internet protocol backbone, XML is used to tag the data, SOAP is used to transfer de data, WSDL is used for describing the services available, and UDDI is used for listing what services are available*). En definitiva, un servicio Web expone funcionalidad a un consumidor, es una URL programable y proporciona mecanismos para invocar operaciones de forma remota (a través de Internet), está basado en estándares Web (HTTP, XML, SOAP, WSDL, UDDI, y más), puede implementarse en cualquier lenguaje y en cualquier plataforma, actuando como caja negra (componentes reutilizable y alquilables).

2.2. Arquitecturas Orientadas a Servicios. SOA

Los servicios Web han puesto de moda las Arquitecturas Orientadas a Servicios (SOA). SOA es una forma de arquitectura para sistemas distribuidos (SD) caracterizada por las siguientes propiedades:

- **Vista Lógica:** El servicio es una abstracción (vista lógica) de los programas, bases de datos, procesos de negocio, etc., definido en términos de lo que hace (llevando a cabo una operación de negocio).

- **Orientación a Mensajes:** El servicio se define formalmente en términos de los mensajes intercambiados entre agentes proveedores y solicitantes, y no está basado en las propiedades de los agentes. La estructura interna del agente (lenguaje de programación, BD, proceso, etc.) se abstrae en SOA. Esto permite incorporar cualquier componente o aplicación a esta arquitectura "decorando" estos componentes con software de gestión y conversión.

- **Orientación a la Descripción:** Un servicio se describe con metadatos procesables. La descripción da soporte a la naturaleza pública de SOA: sólo se incluyen en la descripción

aquellos detalles que se exponen públicamente y son importantes para el uso del servicio. La semántica de un servicio debe documentarse, directa o indirectamente, por su descripción.

- **Granularidad:** Los servicios tienden a usar un pequeño número de operaciones con mensajes relativamente complejos.

- **Orientación a la Red:** Los servicios tienden a usarse a través de la red, aunque este no es un requisito absoluto.

- **Neutral a la Plataforma:** Los mensajes se envían en un formato estándar y neutral a la plataforma, distribuido a través de las interfaces (XML).

En general, SOA y Servicios Web son apropiados para aplicaciones:

- Que deben operar a través de Internet, donde la fiabilidad y la velocidad no se puede garantizar;

- Donde no existe habilidad de gestionar la instalación de forma que todos los solicitantes (clientes) y proveedores se actualicen a la vez;

- Donde los componentes de un sistema distribuido se ejecuten en distintas plataformas y distintos productos;

- Donde una aplicación existente necesite exponerse para ser usada a través de la red y pueda "decorarse" como un servicio Web.

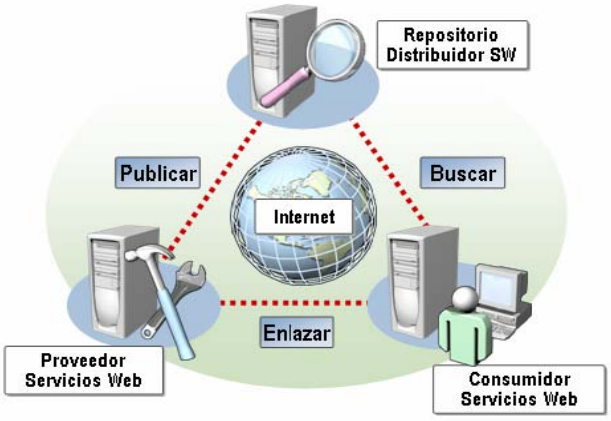


Figura 1
Arquitectura Orientada a Servicios

En términos de *modelos de interacción abstractos* y tal como se aprecia en la figura 1 SOA requiere de las siguientes interacciones:

- Un servicio Web publica su definición (WSDL) en un repositorio / directorio / registro distribuido (UDDI).

- El consumidor del servicio Web busca la definición del servicio en el repositorio.

- El repositorio usa la información de la definición (WSDL) para enlazar con el servicio y enviar peticiones (SOAP) al servicio que ofrece el proveedor de servicios.

2.2.1. Estándares y tecnologías subyacentes. Infraestructura básica

La infraestructura mínima que requieren los Servicios Web se puede definir en términos de:

- Lo que va en "la red": Formatos y protocolos de comunicación
- Lo que describe lo que va en la red: Lenguajes de Descripción de Servicios
- Lo que nos permite encontrar y almacenar dichas descripciones: *Descubrimiento de Servicios*.

2.3. La Pila de las Tecnologías.

Las especificaciones que se han desarrollado para implementar estos mecanismos (en muchas propuestas se presentan como una pila de tecnologías donde las especificaciones superiores hacen uso de las inferiores (ver figura 2) son:

- **HTTP**: (Hypertext Transfer Protocol) HTTP es un protocolo estándar de W3C para la transferencia de documentos en Internet. Los Servicios Web lo utilizan como mecanismo de comunicación. Es un protocolo genérico y sin estado.
- **XML**: (eXtensible Markup Language) lenguaje que deriva de SGML, diseñado para representar y transferir datos estructurados, separa datos de formateo y transformación.
- **SOAP**: (Simple Object Access Protocol, www.w3.org/2002/ws/), ofrece los mecanismos de comunicación básicos para el envío de mensajes en formato XML, permitiendo la invocación remota de servicios. Normalmente funciona sobre HTTP, pero no siempre. SOAP es el *sine qua non* de los servicios Web.
- **WSDL**: (Web Services Description Language, www.w3.org/TR/wsdl), es un formato basado en XML (desarrollado por IBM y Microsoft) para describir de manera formal servicios Web.
- **UDDI**: (Universal Description, Discovery, and Integration, www.uddi.org), es un directorio que contiene un registro/repositorio de descripciones de servicios Web.

Figura 2
La pila de los Servicios Web

2.4. SOAP (Simple Object Access Protocol). El Formato de los Mensajes.

SOAP (en su versión actual 1.2, recomendado por W3C en 2003) define un protocolo que da soporte a la interacción (datos + funcionalidad) entre aplicaciones en entornos distribuidos y heterogéneos, es interoperable (neutral a plataforma y lenguajes de programación, independiente del HW y protocolos). Funciona sobre la infraestructura (estándares) existente en Internet. SOAP define cómo organizar información usando XML de forma estructurada y tipada para intercambiarla entre distintos sistemas.

2.4.1. ¿Qué especifica SOAP?

SOAP especifica:

- Un formato de mensaje para una comunicación unidireccional, describiendo cómo se empaqueta la información en documentos XML.

Un conjunto de convenciones para usar mensajes SOAP para implementar el patrón de interacción RPC, definiendo cómo los clientes pueden invocar un *Procedimiento Remoto* enviando un mensaje SOAP y cómo los servicios pueden responder enviando otro mensaje al llamador.

Un conjunto de reglas que una entidad que procesa mensajes SOAP debe seguir, definiendo en particular los elementos XML que una entidad debe leer y entender, así como las acciones que deben tomar si no entienden el contenido. Reglas de Codificación de los Datos.

Una descripción de cómo se debe transportar un mensaje SOAP sobre HTTP y SMTP. Se definirán "*Bindings*" a otros protocolos de transporte en futuras versiones de la especificación.

2.4.2. El Formato del Mensaje

SOAP intercambia información mediante mensajes. Los mensajes se utilizan como envoltorios que la aplicación utiliza para guardar la información que quiere enviar. Cada envoltorio contiene dos partes: Una cabecera (opcional) y un cuerpo (obligatorio). La cabecera y el cuerpo pueden tener múltiples subpartes en forma de bloques de la cabecera y bloques del cuerpo (ver figuras 3 y 4).

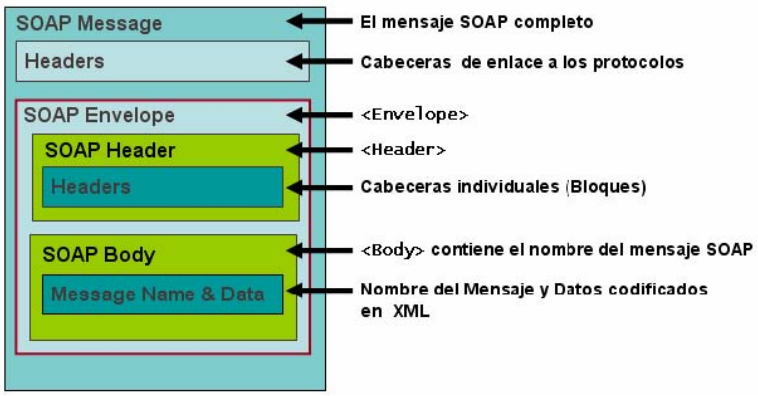


Figura 3
Formato de un Mensaje SOAP

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi=…>
  <soap:Header>
    <WoodgroveAuthInfo xmlns="http://tempuri.org/">
      <Username>string</Username>
      <Password>string</Password>
    </WoodgroveAuthInfo>
  </soap:Header>
  <soap:Body>
    <GetAccount xmlns="http://tempuri.org/">
      <acctID>int</acctID>
    </GetAccount>
  </soap:Body>
</soap:Envelope>
```

Figura 4
Mensaje SOAP

2.5. WSDL (Web Services Description Language). El Lenguaje de Descripción.

WSDL fue creado originalmente por IBM, Microsoft y Ariba (actualmente en la versión 2.0). Tiene el rol y el propósito similar al de los IDL de las plataformas middleware. Un archivo WSDL es un documento XML que describe los servicios Web, en particular sus interfaces. Como característica que lo diferencia de los IDL es que WSDL debe definir los mecanismos de acceso (protocolos) a los servicios Web mientras que IDL no los define en el middleware convencional porque se suponen por defecto. Otra característica diferenciadora es la necesidad de definir (en la especificación) la localización del servicio (puntos finales). La separación de interfaces y enlaces de protocolos, y la necesidad de incluir información de localización permite la definición de especificaciones modulares. WSDL permite definir interfaces más complejas y expresivas permitiendo definiciones de interacciones asíncronas y diferentes paradigmas de interacción, y la posibilidad de combinar o agrupar operaciones.

2.5.1. Estructura de un Documento WSDL

El documento WSDL de un servicio proporciona dos piezas de información básicas:

(1) Una parte o interfaz *abstracta* (independiente de la aplicación)

(2) Una parte *concreta* que define los enlaces a protocolos e información de los puntos finales de acceso al servicio (ver figura 5).

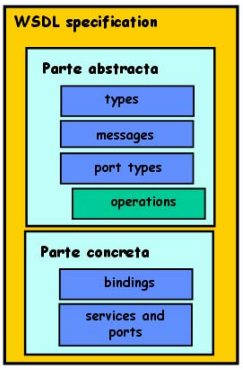


Figura 5
Estructura de un Documento WSDL

La parte abstracta está compuesta por:

- Definiciones de *Port types*: análogos a los interfaces en los IDL. Cada *port type* es una colección lógica de *operations*.

- Cada *operation* define un intercambio simple de *menssages*. Un *message* es una unidad de comunicación representando un intercambio de datos en una única transmisión lógica.

- Un sistema de tipos (*types*) usados por las *operations* (por defecto XML schema).

La parte concreta está compuesta por:

- Definiciones de *Bindings*: se especifica la codificación de los mensajes, y los enlaces a protocolos de todas las operaciones y mensajes definida en un *port type*.

- Definiciones de *Ports*: se especifica en qué dirección (URI) se puede acceder la implementación del *port type*. Definen un punto final (lugar de la red) donde está el servicio.

- Definiciones de *Services***:** definen una agrupación de *Ports*.

De esta forma WSDL se utiliza para describir un Servicio Web en términos de los *mensajes* que *acepta* y *genera*, actúa como *contrato* entre un consumidor (cliente) y dicho Servicio. WSDL puede describir *puntos finales* y sus *operaciones* sin especificar el *formato* de los mensajes o los protocolos de red (Simple Object Access Protocol (SOAP) 1.1, HTTP-GET/POST y MIME) a los cuales el punto final está ligado. En el desarrollo se utiliza como entrada a los compiladores de Stubs y Proxies (ver ejemplo en figura 6).


```
<?xml version="1.0"?>
<definitions name="StockQuote"

    <types>
        <-- XSD for the messages -->
    </types>

    <message name="GetLastTradePriceInput">
        <-- part(s) identifying the messages within the schema -->
    </message>

    <portType name="StockQuotePortType">
        <operation name="GetLastTradePrice">
            <-- grouping messages into logical operations -->
        </operation>
    </portType>

    <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
        <-- specifying how operations map to protocols such as SOAP -->
    </binding>

    <service name="StockQuoteService">
        <port name="StockQuotePort" binding="tns:StockQuoteBinding">
            <soap:address location="http://example.com/stockquote"/>
            <-- providing specific access points for a service -->
        </port>
    </service>

</definitions>
```

Figura 6
Ejemplo de archivo WSDL

2.6. UDDI (Universal Description, Discovery, and Integration). El Repositorio de Servicios.

UDDI, creado originalmente por IBM, Microsoft y Ariba, desde su versión 3 (en 2002) ha pasado a manos de OASIS (www.oasis-open.org) que ha determinado su futuro y extensiones. UDDI se concibió como un Registro de Negocio = *Servicio de Directorio y Nombrado sofisticado*. Especifica un marco para describir y descubrir Servicios Web. UDDI define estructuras de datos y *APIs* para publicar descripciones de servicios en el registro y para consultar el registro en búsqueda de descripciones publicadas. Las APIs de UDDI están especificadas con WSDL y con SOAP Binding, lo que permite acceder a ellas como Servicios Web. La especificación UDDI tiene dos objetivos esenciales: (1) ser un soporte a los desarrolladores para encontrar información sobre servicios Web y poder construir clientes, (2) facilitar el Enlace Dinámico de Servicios Web, permitiendo consultar referencias y acceder a servicios de interés.

2.6.1. Modelo de Información. Estructura de Datos de UDDI

La información en un registro UDDI se almacena en ficheros XML con una estructura jerárquica (ver figura 7). Los elementos de esta estructura son:

- **businessEntity:** es el elemento "top-level", describe un negocio o una entidad que ha registrado un servicio en UDDI. Ejemplos: Departamento de Contabilidad, o Servidor de Aplicaciones Corporativo. Este elemento soporta información estándar tal como nombre, descripción, e información de contacto, así como también información de metadatos (por ejemplo: identificadores y categorías).

- **businessService:** describe un Servicio Web que ha sido expuesto por una entidad de negocio; soporta el nombrado de un Servicio Web y lo asocia con una entidad de negocio y con la información de *binding*. Soporta la asignación de categorías al Servicio Web (industria, productos, códigos geográficos, etc.).

- **bindingTemplate:** describe la información técnica necesaria para enlazar con un Servicio Web en particular. Este elemento soporta el nombrado de un Servicio Web y su asociación con una entidad de negocio e información de binding. La información de binding se describe como un punto de acceso que posee un atributo llamado UrlType utilizado para especificar los siete tipos de puntos de entrada: *mailto, http, Https, Ftp, Fax, Phone, Other.*

- **tModel:** (*Technology Model)*. Estructura de metadatos genérica para representar cualquier concepto o construcción (definiciones de protocolos, ficheros WSDL, XML schemas, Espacios de Nombres, esquemas de categorías, etc.).

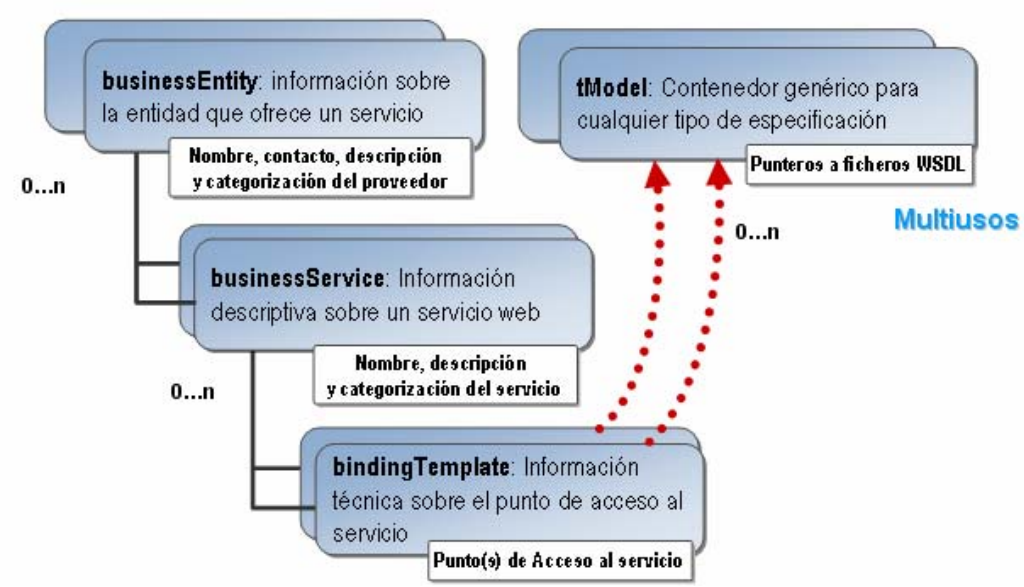


Figura 7
Estructura de Datos UDDI

3. Orquestación de Servicios Web

Las organizaciones que ya están ocupando integración de aplicaciones de empresas (EIA, sigla en inglés), middleware para automatizar procesos de negocios o para integrar ambientes de desarrollo, ya estarán familiarizadas con el concepto de orquestación. Poseen un controlador central que maneja un conjunto de lógicas que facilitan la interoperabilidad entre las distintas aplicaciones. Una implementación común de orquestación es el modelo *hub-and-spoke* que permite que múltiples participantes externos interactúen con un motor central de orquestación.

Uno de los requerimientos manejados para la creación de estas soluciones fue la acomodación de la mezcla de procesos de negocios grandes. Con orquestación, diferentes procesos pueden ser conectados sin tener que volver a desarrollar la solución que originalmente fue automatizada en cada uno de los procesos, por esto, orquestación puede reducir significativamente la complejidad de los desarrollos de soluciones. Ya que orquestación lleva a introducir una nueva lógica de flujos de trabajos, que es abstraída y más fácil de mantener que cuando son soluciones embebidas con componentes individuales.

El rol de la orquestación abarca el desarrollo orientado a servicios. A través del uso de las extensiones que permiten a la lógica de los procesos de negocios ser expresadas a través de servicios, la orquestación puede representar y expresar la lógica del negocio en un estándar. Cuando se construye una solución orientada a servicios, esta provee un significativo atractivo en el control y encubrimiento de la representación de la lógica del proceso a automatizar.

Orquestación más adelante tendrá la capacidad intrínseca de la interoperabilidad buscada para el diseño de servicios y para proveer la potencial integración entre procesos. Un aspecto clave de cómo orquestación está posicionada con SOA es el hecho que orquestación por sí sola existe como servicio. Por lo tanto, construido sobre lógica, la orquestación estandariza la representación del proceso a través de una organización, mientras abordar el objetivo de la supervisión de los servicios de la empresa y promueve la orientación a los servicios.

Una primera especificación de la industria para la estandarización de la orquestación es *Web Services Business Process Execution Language (WS-BPEL)* que también es conocida como *BPEL4WS* o simplemente como *BPEL*.

3.1. Protocolos de Negocios y Definición de Procesos

La lógica del flujo de trabajo que comprende la orquestación puede consistir en numerosas reglas de negocios, condiciones, y eventos. Colectivamente, estas partes de una orquestación establecen un protocolo de negocio que define cómo los participantes pueden llegar a completar los procesos de negocio. El detalle de la lógica del flujo de trabajo está encapsulado y expresado por una orquestación que está contenida con una definición del proceso.

3.2. Servicio de Procesos y Socio de Servicios

Identificado y descrito con una definición de proceso están los participantes permisibles del proceso. Primero, el proceso en sí es representado como un servicio, resultando en un servicio de procesos (ver figura 8).

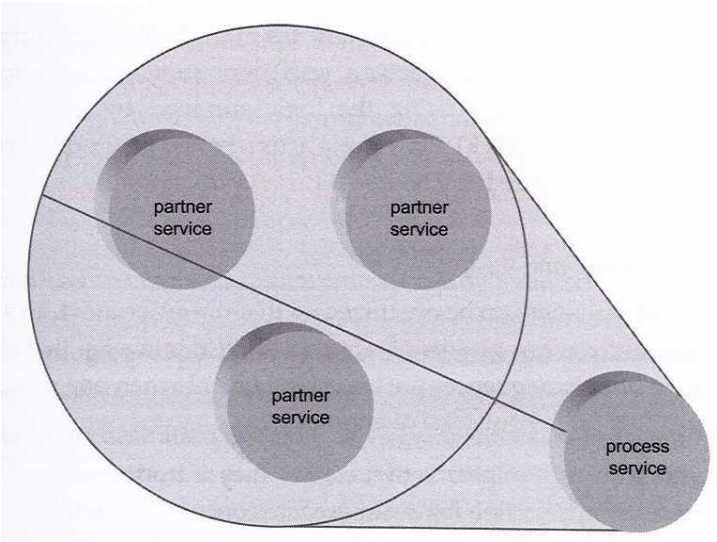


Figura 8
Servicio de Procesos

Otro servicio permitido para interactuar con el servicio de proceso está identificado como socio de servicios o *partner links*. Dependiendo sobre la lógica del flujo de trabajo, el servicio de proceso puede ser invocado por un socio de servicio externo, o puede ser invocado por otro socio de servicio (ver figura 9).

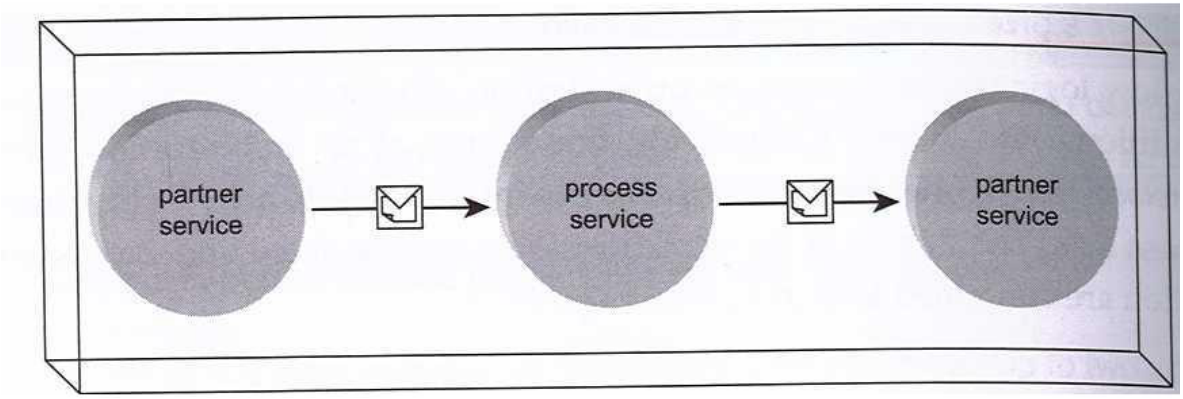


Figura 9
Flujo de Trabajo

3.3. Actividades Básicas y Actividades Estructuradas

BPEL4WS analiza la lógica del flujo de trabajo en una serie de actividades primitivas predefinidas. Las *Actividades Básicas* (recieve, invoke, reply, throw, wait) representan fundamentalmente acciones en el flujo de trabajo las cuales puede ser ensamblada usando la lógica provista por las *actividades estructuradas* (squence, switch, while, flow, pick).

3.4. Secuencias, Flujos y Acoplamientos

Actividades básicas y estructuradas pueden ser organizadas de modo que el orden en el cual ellas se ejecuten sea predefinido. Una *secuencia* junta grupos de actividades relacionadas en una lista que determina un orden de ejecución de la secuencia. Las secuencias son especialmente usadas cuando una pieza de la lógica de la aplicación es dependiente de las salida de otra.

Flujos también contiene grupos de actividades relacionadas, pero ellas introducen diferentes requerimientos en la ejecución. Pedazos de lógica de la aplicación puede ser ejecutada concurrentemente con un flujo, significa que éste no es necesariamente un requerimiento de un grupo de actividades que espera a que otras terminen. Sin embargo, el flujo en sí no termina hasta que todas las actividades encapsuladas en él hayan sido procesadas por completo. Esto asegura una forma de sincronización entre la lógica de la aplicación residente en los flujos individuales.

Acoplamientos es usada para establecer dependencias formales entre actividades que son partes de un flujo. Antes de que una actividad esté completamente terminada, se debe estar seguro que cualquier requerimiento establecido desde un acoplamiento externo ya fue respondido. Similarmente, antes de que cualquier actividad acoplada pueda empezar, es necesario que sea satisfecho cualquier otro requerimiento de acoplamientos hecho con anterioridad. Reglas provistas por el acoplamiento son siempre referidas como *dependencias de sincronización*.

3.5. Orquestación y Actividades

Una actividad es un término genérico que puede ser aplicado a cualquier unidad de lógica de un trabajo completado por una solución orientada al servicio. Este es el alcance de una orquestación, por lo tanto, puede ser clasificado como un complejo, y muy probablemente, actividades duraderas.

3.6. Orquestación y Coordinación

Orquestación, es representado por BPEL4WS, puede utilizar completamente el manejador de contexto del cuadro de WS-Coordination que está incorporado en el coordinador de tipos de WS-BussinesActivity. Esta especificación define el diseño de protocolos de coordinación para el complejo soporte, actividades duraderas.

3.7. Orquestación y SOA

La lógica del proceso de negocio es la raíz de la automatización de las soluciones. Orquestación provee un modelo automatizado donde la lógica de procesos es centralizada con todo aun extensible y componible. A través de uso de la orquestación, el desarrollo de la soluciones orientadas al servicio llegando a ser intrínsecamente extensibles y adaptables. Orquestación

típicamente establece un punto en común de integración para otras aplicaciones, las cuales hacen una orquestación implementando una llave de integración.

Estas cualidades incrementan la agilidad organizacional porque:

- La lógica del flujo de trabajo encapsulado por una orquestación puede ser modificado o extendido en un lugar centralizado.

- Posicionando una orquestación centralizada pude facilitar significativamente la mezcla de procesos de negocios por abstracción del unificador que une la correspondiente solución automatizada.

- Por establecimiento potencialmente de interacción de arquitecturas de gran escala orientada al servicio. Orquestación, sobre un nivel fundamental, puede soportar la evolución diversificada de la empresa.

Orquestación es el ingrediente clave para la realización de una organización que contiene varias aplicaciones basadas en plataformas computacionales dispares. Avances en el middleware permiten que los motores de orquestación se vuelvan completamente integrados al desarrollo orientado al servicio.

Para muchos desarrollos, orquestación se ha convertido en el corazón de SOA.

4. Herramientas de Orquestación de Servicios Web

Si bien no existen herramientas que soporten solamente orquestación, aquí mostraremos cuatro herramientas que de alguna manera tienen incluido el concepto de un orquestador de servicios Web dentro de ellas.

De todas las herramientas que mostramos aquí, una es gratuita y las otras tres son pagadas. De estos últimos, uno es de una gran empresa de Base de Datos, otro es de una empresa que lleva años en el desarrollo de software, y el tercero de una empresa que está empezando el desarrollo de este tipo de software.

Procederemos a entregar una breve descripción de cada una según las características entregadas por los distintos proveedores de dichas herramientas.

4.1. Intalio BPMS

Intalio/BPMS 4.0 fue construida entorno a tres ideas principales: soportar los últimos estándares de la industria en BPM, ofrecer un acercamiento al desarrollo sin código (a través de una interfaz gráfica) y proveer una forma de desarrollar ciclos de vida en un click.

4.1.1. Intalio/Designer

Intalio/Designer es un ambiente integrado basado en Eclipse para el desarrollo de los procesos del negocio de BPMN. Los analistas del negocio y los ingenieros de software pueden hacer uso de Intalio/BPMS porque es una manera fácil para tender un puente entre el negocio y las tecnologías de la información (TI). También, puesto que ofrece el desarrollo sin código (sólo a través de interfaz gráfica) y el despliegue en un click, no se tiene que ser un especialista de J2EE o un experto de XML para utilizarlo, y si se es uno de todos modos, encontrará en Intalio/Designer un reforzador formidable de la productividad.

4.1.2. Intalio/Server

Intalio/Server es un servidor de proceso de BPEL 2.0 nativo basado en J2EE. Debido a que es una arquitectura referente a la nueva tecnología de la integración del negocio de Java (JBI), puede ser desplegado en forma virtual en cualquier servidor del uso de J2EE. El servidor de proceso fue diseñado desde el comienzo para ser desplegado en una grilla de 1.000 servidores por el Ministerio de Energía de los EEUU así que, si el proyecto requiere la ejecución de procesos de alto rendimiento, esta puede ser una buena solución.

4.1.3. Intalio/Workflow

Intalio/Workflow es un conjunto humano integrado por el workflow basado en las nuevas extensiones de BPEL4People y compatible con cualquier portal de JSR 168. Ofrece una puesta en práctica basada en AJAX de XForms; da a los participantes del workflow una experiencia productiva y de acoplamiento con el usuario, mientras que es compatible con cualquier navegador Web. La suite del workflow es accionada directamente por el servidor de procesos, permitiendo que se desarrollen los patrones del workflow.

4.2. Oracle Fusion Middleware

Oracle Fusion Middleware es un conjunto de productos de última tecnología, probados por los clientes de Oracle y basados en estándares de la industria. Este Middleware incluye herramientas y servicios J2EE, servicios de integración, inteligencia de negocios, colaboración y manejo de contenido.

Esta familia de productos integrada y completa ofrece todos los elementos necesarios para el desarrollo, implementación y administración de aplicaciones en una Arquitectura Orientada a Servicios (SOA, Service Oriented Architecture). Oracle Fusion Middleware está basada en una arquitectura compatible (en inglés, hot-pluggable) que permite aprovechar la inversión actual en sistemas o tecnología. Finalmente, como la disponibilidad de los sistemas es fundamental para el negocio, las cualidades de redundancia de la tecnología de fondo de Oracle Fusion Middleware minimizan la interrupción de caídas del sistema, sean estas planificadas o no.

Estos atributos únicamente disponibles en Oracle Fusion Middleware, le permiten:

- Optimizar y agilizar las operaciones del negocio y de TI.
- Mejorar la precisión de las decisiones de negocio, facilitando que ocurran en el momento oportuno.
- Resguardar la seguridad de la información y cumplimiento de las políticas de la organización minimizando la interrupción del negocio.
- Esta solución es compatible con lo que la empresa tiene en este momento, así que no es necesario comenzar de nuevo. Más de 26.000 clientes a nivel mundial están mejorando su negocio integrando Oracle Fusion Middleware en su infraestructura tecnológica.
- Oracle Fusion Middleware consta de varias partes. En este caso la que nos importa estudiar por sus características es Oracle SOA Suite.

4.2.1. Oracle SOA Suite

Es un conjunto de software compatible para la construcción, desarrollo y manejo comprensible de una arquitectura orientada a servicios. Esto incluye el desarrollo de aplicaciones orientadas a servicios, la integración de aplicaciones y sistemas de tecnologías orientadas a servicios, y orquestación de procesos de servicios de sistemas y workflow humano. Esto los conecta en forma heterogénea a la infraestructura de las tecnologías de la información y deja preparada a la empresa para que vaya adoptando en forma incremental un SOA. Los componentes de esta suite benefician desde las capacidades comunes incluyendo el desarrollo y manejo de los modelos, utilitarios, seguridad en todas las capas y manejo unificado de la metadata.

Mejora la capacidad de una organización de predecir el cambio cercano mejorando su visibilidad a los eventos en el ambiente del negocio en tiempo real y poder responder al cambio permitiendo desarrollar y optimizar procesos del negocio rápidamente. Lo simplificado del ambiente de las tecnologías de la información permite tomar las provisiones necesarias, desplegándolo, supervisándolo, y manejándolo como una sola infraestructura cohesionada. Influencia inversiones existentes siendo modular, abierto, y extensible; puede ser adoptado en un ambiente heterogéneo

sin necesitar quitar o sustituir sistemas existentes así como también se puede insertar de una manera incremental.

Este consiste de:

- BPEL Process Manager para componer servicio en base a los procesos de negocios;
- Un Monitor de Actividades de Negocios (BAM, Business Activity Monitor) solución para tener en tiempo real la visibilidad de la operación y el rendimiento de los servicios de los procesos de negocios;
- Un motor de reglas de negocios para capturar y automatizar las políticas del negocio;
- Un bus de Servicio para la empresa multi-protocolo (ESB, Enterprise Service Bus) para conectar las aplicaciones y redirigir los mensajes;
- Un Manejador de servicios Web y solución de seguridad para reafirmar la autenticación y autorización de las políticas de seguridad en los servicios;
- Un Registro de servicios para el descubrimiento y manejo de los ciclos de vida de los servicios; y
- Un Ambiente integrador de servicios (ISE, Integrated Service Environment) para desarrollar, eliminar errores, juntar y visualizar los servicios.

4.3. Agila

Apache Agila es un nuevo aporte de la asociación de software de Apache, consiste en un motor de BPM liviano y servicios auxiliares.

Agila ha sido aceptada como parte de Apache Jakarta PMC para la inclusión en el proyecto Jakarta. Una vez que los requerimientos para la incubación estén completos, la comunidad y el código serán movidos hacia Jakarta.

Agila esta centrada alrededor de Manejo de Procesos de Negocios (BPM, Business Process Manager), Workflow y Orquestación de servicios Web. Está compuesta de módulos especializados: Agila BPM y Agila BPEL.

4.3.1. Agila BPM

Es básicamente un manejador de tareas y usuarios que tienen que completar dichas tareas. Es un componente bastante liviano y flexible de workflow.

4.3.2. Agila BPEL

Es una solución que cumple el estándar de BPEL de orquestación de servicios Web. A continuación una lista de cosas para qué es bueno y para qué no es bueno BPEL según agila:

- BPEL no es workflow: no hay abstracciones explícitas para la gente, los papeles, los artículos del trabajo, o los inboxes en BPEL (entre otras cosas).
- BPEL no es tampoco BPM: ninguno de los datos especificados modelan los procesos para la medición, divulgación, o el manejo.
- BPEL no es integración: no hay ayuda explícita para la transformación, la interpolación semántica, o los protocolos específicos.
- BPEL no abarca todo: hay algunos patrones que son difíciles de modelar con BPEL.
- BPEL (es decir, una puesta en práctica) abstrae del campo común, preocupaciones del nivel de la ejecución del proceso del negocio.
- BPEL proporciona una interfaz del nivel de despliegue para las herramientas de alto nivel.
- La innovación de tener un lenguaje común no crea ventaja competitiva sostenible.
- Definir los procesos de organización alrededor de herramientas que utilizan BPEL, es de importancia suprema para el papel y el éxito de la arquitectura orientada al servicio.

4.4. Fuego

FuegoBPM es una plataforma comprensiva, avanzada para la gerencia del software de procesos de negocios. Fuego ayuda al negocio aumentando la eficiencia operacional, reduciendo los costos y aumentando los beneficios, mediante el mejoramiento de los workflow y falibilidad de los sistemas SOA/BPM. FuegoBPM automatiza y optimiza los procesos de negocios para ayudar a los clientes a reducir costos, mejorar su servicio y ser más ágiles en una fracción de los costos de las soluciones tradicionales. Muchos de sus clientes implementaron los procesos en toda su magnitud en un periodo de entre 30 y 60 días.

4.4.1. Características y Ventajas

Con 6 módulos muy cohesionados, la suite FuegoBPM automatiza, maneja y optimiza de principio a fin el ciclo de vida de un proceso.

4.4.2. Desarrollo del Proceso

El FuegoBPM Designer y Studio completan todos los aspectos de la creación y desarrollo de un proceso. Un modelo analítico de un proceso de negocio, define reglas del negocio y los indicadores dominantes del funcionamiento (KPI, Key Performances Indicador), simulándolos y probándolos. Los desarrolladores de tecnologías de la información (TI) usan estas mismas herramientas para integrar rápidamente cualquier aplicación o sistema dentro de un proceso. La administración de procesos y la seguridad de la empresa están aseguradas por su incomparable funcionalidad y robusta arquitectura. FuegoBPM suite reduce el ciclo de desarrollo para aplicaciones de procesos en un 70% o más.

4.4.3. Ejecución de Proceso

El Fuego Portal genera automáticamente el ambiente de trabajo del usuario para cada participante del proceso, basado sobre su papel definido en la organización. El Fuego Server orquesta todos los recursos – gente, organizaciones, aplicaciones y sistemas – haciendo cumplir reglas de negocio y manejando todos los recursos en su secuencia apropiada para asegurar el manejo sin defectos de la ejecución, de la escalabilidad y de la excepción. La suite FuegoBPM aumenta perceptiblemente la productividad del usuario y mejora más aun el porcentaje de los niveles de proceso, críticos al éxito de cualquier solución de BPM.

4.4.4. Análisis y Optimización de Procesos

El Fuego Manager y Dashboard es capaz de entregar análisis en tiempo real e histórico, así como también reportes. Monitoreos de procesos en tiempo real, manejo de escalabilidad y excepciones asegura que los problemas son aislados y solucionados rápidamente, a través de su *easy-to-create*, proceso intuitivo del dashboard. La tecnología incomparable de la optimización de Fuego ayuda al funcionamiento real del mapa contra las KPI definidas para el proceso, asegurando el mejoramiento continuo y en curso del proceso

4.4.5. Tecnología y Estándar Soportado.

El software de Fuego esta basado en estándares abiertos, independientes de la plataforma

Funciona sobre: Plataforma Microsoft .Net, cualquier aplicación de servidor J2EE-Compliant o sólo.

Soporta: Todos los estándares de procesos como BPEL y/o XPDL (para ejecución de procesos), BPMN (para notación de modelamiento), etc.

Conectores para: Java, Ensamblador .Net, Corba/IDL, EJB, JDBC, HTML, XML, XML Schemas, automatización (COM+/DCOM), SMTP, JMS, JNDI, Web Services (WSDL), y muchos más.

5. Elementos a Tener en Consideración en la Evaluación.

Al momento de tomar la decisión de qué herramienta para orquestación implementar, se pueden generar variadas expectativas con respecto a lo que el mercado ofrece. Lo más importante es poder saber qué cosas son significativas al momento de decidir qué herramienta de orquestación implementar. A continuación presentamos un listado de las cosas a considerar y que sirvan para justificar una decisión.

- **Soporte de Procesos de Negocios**: Aunque el tema a ver es Orquestación de Servicios Web, no se puede dejar de lado el hecho de para qué estamos buscando dicha orquestación. Como la orquestación queda en parte al servicio de los procesos de negocios, toda herramienta que posea un orquestador de servicios Web debe poseer de alguna manera un soporte para los procesos de negocios.

- **Desarrollo a través de código**: Si bien puede parecer excesivo el pedir el desarrollo a través de código, hay que hacerlo ya que con el auge de las herramientas de programación a través de interfaces gráficas este recurso es cada vez más dejado de lado. Pero la importancia es alta al momento de tener que dar soporte y hacer las cosas de forma muy transparente. Si bien muchas veces es más complejo, no deja de tener importancia al momento de ver las alternativas en la programación.

- **Desarrollo a través de interfaz gráfica**: Este tipo de desarrollo cada vez toma mayor importancia debido a que le da mayor facilidad al usuario común para entender lo que está haciendo y puede ver de mejor manera las cosas que está desarrollando. Si bien no es la manera más acostumbrada de desarrollar, de a poco este método ha ido ganando terreno en los usuarios menos especializados lo que abre un campo muy grande al desarrollo.

- **Soporte de BPEL distintas versiones**: Si bien casi todas las herramientas dan soporte BPEL, hay que tener en consideración todas las variantes de BPEL como versión 2.0 y BPEL4WS, que hoy en día son los estándares más importantes. Esto hace que si bien existe un estándar, dicho estándar no sea muy estricto y que varias compañías han desarrollado por su parte su propio estándar basado o no en versiones de BPEL.

- **Compatibilidad con elementos actuales**: Es importante que al implantar las tecnología de orquestación de servicios Web no se tenga que realizar mayores cambios internos en la empresa, ya que estos siempre generan de alguna manera un trauma, tanto por el tiempo invertido, como por los cambios en el fondo y la forma de trabajar que puede sufrir la empresa. Por eso es de vital importancia la compatibilidad de las herramientas con los elementos que se posea en la empresa para minimizar el trauma a sufrir por la empresa.

- **Interfaz Gráfica**: Si bien hemos hablado del desarrollo a través de una interfaz gráfica, también es bueno recordar que para la administración y manejo de las herramientas que se puedan tener, es bueno tener una interfaz amigable y simple, sin dejar de lado las interfaces de modo consola que sirven para los casos más complejos.

- **Elementos de software**: Así como estamos evaluando los elementos necesarios para poder comparar los distintos programas que poseen orquestación de servicios Web,

también se debe evaluar los elementos mismos del software como son instalación, puesta en marcha, necesidades de hardware para su uso, etc. Si bien esto no es el punto el principal de todas maneras hay que hacer alguna referencia al momento de la evaluación de la herramienta.

6. Generación de Tabla de Evaluación

Los elementos que hemos definido nos dan un pauta de las cosas que se consideran importantes evaluar en los distintos programas que poseen orquestación, pero siempre es bueno tener una forma simple de evaluar dichos elementos, por eso mismo definiremos una lista más detallada de los elementos a tener en consideración. Dicha lista tendría elementos como: soporte BPM, interfaz gráfica para BPM, carga de archivos BPM, generador de archivos WSDL, generador de interfaz Web, generador de procesos, generador gráfico de procesos, generador de archivos XML, motor de ejecución de procesos, soporte BPEL, soporte BPEL 2.0, soporte BPEL4WS, ejecución de BPEL, creación de XML, interfaz gráfica para BPEL, interacción con servicios Web existentes, interacción con otros servicios existentes, claridad en instalación, facilidad en la instalación, necesidad de paquetes secundarios para la ejecución, entendimiento de la interfaz.

Al tener este listado podemos generar una tabla con dichos elementos para verlos de manera más ordenada y que sea más simple al momento de tener que comparar las distintas herramientas. Los elementos se agruparán de la siguiente manera:

6.1. Herramienta

6.1.1. Versión

Es muy importante saber a que versión del programa se le están haciendo las pruebas, aunque esta no una evaluación, es información muy valiosa que no hay que dejar de lado y por eso la agregamos.

6.1.2. Instalador

Ver si el instalador del programa es amigable o no, así mismo si funciona o no de manera correcta al momento de ejecutar la instalación. Aunque estos elementos parezcan simples puede hacer la diferencia entre una herramienta y otra.

6.1.3. Dependencias

Muchos de estos programas no funcionan por sí solos, sino que requieren una serie de programas que les presten servicio y apoyo para su funcionamiento. Esto puede significar incurrir en mayores gastos de espacio, recursos y dinero al momento de instalar alguno de estas herramientas.

6.1.4. Ejecución

Aunque parezca paranoico el hecho que la ejecución del programa sea correcta es algo a tener en consideración al momento de querer evaluar un programa, ya que no por tener una instalación correcta la ejecución debe serlo.

6.1.5. Ayuda en Línea

La ayuda en línea de los programas hoy en día es el medio más ocupado para poder resolver los problemas por parte de los usuarios, por lo mismo es muy importante que dicha ayuda en primera instancia exista y que sea realmente un apoyo.

6.1.6. Ejemplos

Los ejemplos son los primeros acercamientos que tiene el usuario al momento de tratar de usar un programa y debido a esto es importante que todo programa traiga los mejores ejemplos posibles, para hacer dicho acercamiento lo más asequible para el usuario.

6.2. Procesos de Negocios

6.2.1. Interfaz

Para el desarrollo de los procesos de negocios se puede o no poseer una interfaz gráfica, pero siempre debe poseer algún tipo de interfaz.

6.2.2. Generación de Código

Debe ser capaz de generar un mensaje SOAP para poder comunicarse con elementos en un UDDI para interactuar con los servicios y servidores. Básicamente lo que se espera es que al poder generar a través de una interfaz un procesos de negocio, que dicha interfaz sea capaz de transformar en una serie de código entendible tanto por los servidores como por los servicios, ya sea que este código pase por la generación de elementos como diagramas de procesos de negocios, esquemas XML, archivos WSDL, formularios Web, documentación y otros.

6.2.3. Consola

Consiste en tener la opción de poder trabajar a través de una interfaz básica como es una consola de texto para los casos más complejos. Esta consola generalmente se ocupa en momento de emergencia para hacer carga y descarga de modelos en forma serializada y en situaciones en general.

6.3. BPEL

6.3.1. Soporte de Versiones

En general todas estas herramientas traen soporte para BPEL, pero hoy en día existen varias versiones de BPEL como son BPEL 2.0 y BPEL4WS. Aunque estas se parecen no siempre son iguales por lo que es importante saber que versiones soporta y generan cada uno de estos programas.

6.3.2. Generación de Código

Es importante saber si es capaz de generar el código BPEL para poder ser exportado para otras aplicaciones. Así mismo que dicho código sea legible o interpretable fácilmente por un usuario experto y otras aplicaciones.

6.3.3. Consola

Consiste en tener un opción de trabajar a través de una interfaz básica para hacer la carga y descarga de programas en caso de emergencia o de hacer una carga masiva de *scripts*.

6.3.4. Debugger

La idea es que posea un programa que permita hacer un seguimiento paso a paso, reconocer errores y manejar el código de los procesos hechos en BPEL.

6.4. Motor ESB (Entreprise Services Bus)

Este motor permite la ejecución de los distintos procesos modelados o cargado con las herramientas de modelamiento. Éste incluye el motor de orquestación ya que éste en general no existe por si sólo.

6.4.1. Interfaz

Todo motor de ESB (Entreprise Service Bus) debe tener una interfaz de algún tipo. Por lo general traen una interfaz Web la cual permite ver que procesos se están ejecutando y realizar ciertas operaciones sobre dichos procesos.

6.4.2. Operaciones

Consiste en ver cuales son las capacidades que se tiene para realizar ciertas operaciones como activar, desactivar, eliminar y agregar sobre los procesos que se están ejecutando en ese momento sobre el motor.

6.4.3. Estadísticas

Siempre es importante poder saber cuales son las estadísticas del motor, en el sentido de poder saber que procesos son los más ejecutados por que grupo de usuarios, cuanto tiempo toma en realizar tal y cual proceso, etc.

6.4.4. Compatibilidad

La compatibilidad del motor es la capacidad de ejecutar elementos que no han sido desarrollados con los programas que se entregan con o para dicho motor, sino que por otros programas o simplemente que han sido desarrollados siguiendo un estándar. Así también consiste en ver su capacidad de relacionarse con otros motores.

6.4.5. Comunicaciones

Hablamos de la capacidad que tiene el motor de comunicarse con otras aplicaciones, tanto del mismo fabricante como de fabricantes distintos. Esto incluye la posibilidad de comunicarse con distintos servicios y a distintos protocolos.

6.5. Tabla

Aquí presentamos la tabla a llenar en la evaluación de los programas.

Programa	
Versión	
Instalación	
Dependencias	
Ejecución	
Ayuda en línea	
Ejemplos	
Procesos de Negocios	
Interfaz	
Generación de código	
Consola	
BPEL	
Soporte de versiones	
BPEL 2.0	
BPEL4WS	
Generación de código	
Debugger	
Consola	
Motor ESB	
Interfaz	
Gráfica	
Texto	
Operaciones	
Estadisticas	
Compatibilidad	
Comunicaciones	

7. Aplicación a herramientas

Se Presentan los resultados que tuvo la aplicación de la tabla realizada en las distintas herramientas seleccionadas.

Dichos resultados fueron obtenidos en una situación de laboratorio. Se ejecutaron los ejemplos que se encuentran en los anexos y los resultados fueron comparados contra ellos.

De la misma forma los programas se ejecutaron en dos computadoras distintas.Un Athlon AMD 2400+ con 2GB en RAM y un Notebook HP nc4010 con la misma cantidad de RAM.

7.1. Intalio BPMS

De la empresa Intalio, posee tres componentes que son Intalio Designer, Intalio Server e Intalio Workflow. Partner de Apache y OpenSource que decidió entregar en forma gratuita su herramienta desde este año.

Programa	
Versión	4.3.2
Instalación	Bastante simple por hay que seguir muy expresamente el manual.
Dependencias	Necesita tener instalado jre1.4 al menos
Ejecución	Un poco escondido ya que no deja ningún icono en el escritorio
Ayuda en línea	Excelente muy simple de encontrar y seguir
Ejemplos	Buenos ejemplos muy bien explicados
Procesos de Negocios	
Interfaz	Muy buena interfaz, no es muy intuitiva en primera instancia pero luego es bastante simple de seguir
Generación de código	Genera un código fácil de leer y cumple estándar
Consola	No se encontró, aunque los fabricantes acusa tenerla
BPEL	
Soporte de versiones	
BPEL 2.0	Soporta completamente
BPEL4WS	Soporta completamente
Generación de código	Genera un código simple y legible. Cumple estándar BPEL 2.0
Debbuger	Tiene y funciona a través de la interfaz gráfica
Consola	Dice poseerla pero no pudo ser encontrada.
Motor ESB	
Interfaz	
Gráfica	Posee y es fácil de usar, bastante intuitiva
Texto	Dice poseerla no pudo ser encontrada
Operaciones	Permite realizar variadas operaciones sobre los procesos en ejecución como son agregar, quitar detener e iniciar.
Estadísticas	Entre estadísticas de uso y tiempo de ejecución
Compatibilidad	Con varias bases de datos pero no se logro conectar a otro servidor Web.
Comunicaciones	Buenas comunicaciones con otros servicios.

7.2. Oracle SOA Suite

De la empresa Oracle y parte del paquete Oracle Fusion Middleware. Posee cinco componentes que son Oracle BPEL Process Manager, Oracle Web Services Manager, Oracle Bussines Rules Engine, Oracle Bussines Activity Monitor, Oracle Entreprise Service Bus y Oracle JDeveloper. Si bien se puede descargar de forma gratuita para realizar cualquier desarrollo de tipo comercial hay que comprar un licencia.

Programa	
Versión	10.1.3.1
Instalación	Produce variados errores por lo que hay intentar varias veces la instalación para que ocurra en forma correcta
Dependencias	Depende una serie de paquetes de Oracle como son la base de datos y JDeveloper que dice traerlo pero hay que descargarlo a parte. También depende de java (jre1.4).
Ejecución	Se ejecuta correctamente
Ayuda en línea	Hay pero muy difícil de encontrar, pero existe en sitio de Oracle
Ejemplos	Posee sólo un ejemplo de cómo hacer las con dichos paquetes.
Procesos de Negocios	
Interfaz	No tiene por si sólo hay que bajar la aplicación JDeveloper que dice traer pero hay realmente hay que bajarla del sitio de oracle y se integra si inconvenientes
Generación de código	Genera código legible y utilizable
Consola	No posee
BPEL	
Soporte de versiones	
BPEL 2.0	Soportada completamente
BPEL4WS	Soportada completamente
Generación de código	Genera código limpio y legible
Debugger	Posee un excelente debugger
Consola	Dice tener pero no se pudo encontrar
Motor ESB	
Interfaz	
Gráfica	Muy buena interfaz, muy legible y fácil de utilizar
Texto	No se encontró
Operaciones	Posee operaciones de carga, descarga, activación, desactivación
Estadísticas	De acceso por usuario, tiempo de proceso, utilización.
Compatibilidad	No es muy compatible con otros motores.
Comunicaciones	Buena comunicación con otros servicios.

7.3. Agila

Producto gratuito desarrollado por el grupo de Apache y que aún está en etapa de desarrollo por lo que sólo se puede obtener las versiones de desarrollo a través del portal de Apache. No es recomendable aún para uso ya que tiene muchas fallas y errores. Se pone aquí como una alternativa que se espera poder tener algún día en forma gratuita e integrada a uno de los servidores Web más ocupados a nivel mundial.

Programa	
Versión	481329 (versión de desarrollo)
Instalación	Compleja ya que hay que compilar la aplicación
Dependencias	Subversion + java (jre 1.4)
Ejecución	Simple no necesita mayores elementos
Ayuda en línea	No hay
Ejemplos	No hay
Procesos de Negocios	
Interfaz	Tiene pero falta desarrollo
Generación de código	Funciona dentro de las funcionalidad de la interfaz
Consola	Si existe y permite cargar el código
BPEL	
Soporte de versiones	
BPEL 2.0	Soporta completamente
BPEL4WS	Soporta completamente
Generación de código	Funciona dentro de las funcionalidades de la interfaz
Debugger	No hay
Consola	Si posee y permite cargar los archivos
Motor ESB	
Interfaz	
Gráfica	Posee una bastante intuitiva pero falta desarrollo
Texto	Posee una de difícil uso
Operaciones	Las mínimas como carga y descarga de procesos
Estadisticas	No encontradas
Compatibilidad	No encontradas
Comunicaciones	Si con algunos servicios

7.4. Fuego

Existían muchas expectativas sobre este programa lamentablemente durante este año dicho producto fue vendido a otra empresa. La empresa que compro Fuego es BEA y lo incorporo a su herramienta SOA para potenciarla y mejorarla, dando como resultado una nueva herramienta llamada BEA Aqualogic BPM. Dicha herramienta salía al mercado casi al momento de la entrega de este informe por lo que no pudo ser evaluada.

8. Conclusiones

Si bien aquí se buscaba ver el funcionamiento de las herramientas de orquestación nos topamos con el hecho que dichas herramientas no existen por si solas, si no más bien en conjunto con otras herramientas para el apoyo de SOA (Service Oriented Architecture) en las empresas. Por lo que se decidió evaluar dichas herramientas que poseen mayores funcionalidades de las que se esperaban evaluar en una herramienta de orquestación.

Siempre se ha querido poder tener medios de comprar la eficiencia, eficacia, utilidad, manejo y simplicidad de una herramienta de software. En esta tesis se ha buscado dar un paso a esto, quizás sin llegar a entregar un herramienta final de comparación debido a que las herramientas aquí presentadas, si bien son comparables en algunos aspectos, en otros no lo son, y son estos últimos los que hacen grandes diferencias entre las distintas herramientas. De todas formas aquí se intento buscar los elementos comunes que poseen dichas herramientas para poder generar una comparación cualitativa de las distintas herramientas que den un panorama más claro al momento de decidir cual herramienta escoger.

Al momento de revisar las herramientas las expectativas sobre sus cualidades siempre son muchas, aunque se estudiaron las capacidades que informa el fabricante, siempre existieron expectativas mayores en base a los estándares prometidos y al hecho que son herramientas basadas en tecnologías que han ido surgiendo con tiempo. Ahora bien si no se cumplieron todas las expectativas hechas, las herramientas evaluadas cumplieron en gran parte con las cosas que prometen, como son mejoras en interfaz y manejo para las personas más inexpertas, aunque esto fue en desmedro de las antiguas consolas de texto en muchos casos y generan lo que se puede considerar un abuso en la interfaz Web para elementos de administración de una herramienta, quizás aquí es donde hay que cuestionarse que hacer en el caso de que la aplicación genera un falla tal que la única manera de intervenir la máquina donde se aloja es mediante la consola de ésta.

Si bien se espera que las tecnologías de Orquestación de Servicio Web permitieran la interacción entre los sistemas ya desarrollados y los nuevos sistemas que se están desarrollando con las herramientas, se ve que la brecha entre los elementos desarrollados con los antiguos estándares del Web, que no consideraban elementos como SOAP, WSDL y UDDI son muy difícil de integrar y su integración sigue siendo a través de complejos *scripts* hechos en lenguajes de ejecución sobre los servidores para poder obtener la información que se necesita de dichas páginas o para poder dar valor al campo requerido en las mismas. Por eso podemos decir que la tarea de compatibilidad si bien esta hecha entre herramientas desarrolladas bajo las mismas tecnologías, no podemos decir que exista una compatibilidad hacia los antiguos estándares como eran en un principio el HTML en su expresión más pura.

Las herramientas evaluadas son completas en muchos sentidos, ya que traen sus propios elementos de publicación Web, repositorios de publicación, motores de ESB y elementos que permiten generar código BPEL, WSDL y demás, según corresponda tanta para el uso propio, como para la exportación de dichos códigos. Ahora bien existen casos en los cuales algunos de estos elementos ya existen dentro de las instituciones que desean adoptar estas herramientas por lo que es necesario que dichas herramientas se integren a los elementos ya existentes. Si bien dicha integración existe y las herramientas traen módulos para realizar la integración con

elementos que no son los desarrollados por las mismas empresas que ofrecen la herramienta, dichos módulos en muchos casos no son simples de instalar o simplemente no funcionan correctamente. Así tenemos que las herramientas si bien funcionan correctamente y cumplen lo prometido esto es siempre y cuando sea bajo las condiciones y con los programas que ofrecen ellas y no con los de otros fabricantes.

Hoy en día, se habla de estándares y de su generación en el área de las comunicaciones y servicios, pero al momento de ver si las herramientas cumplen o tienen algún estándar asociado que no sea en los archivos que genera sino en la ejecución, orden e iconográfica para realizar los modelos, se encuentra que eso no existe. Esto hace que el usuario finalmente prefiera una herramienta a otra por un tema de cautiverio de conocimientos, ya que la adaptación a uso de una nueva herramienta es muy lento y en el mundo de hoy lo que más se necesita es agilidad al momento de realizar los cambios. Si bien no es un elemento de gran trascendencia al momento de estudiar una herramienta, en el momento de querer comprar una herramienta nueva para implementarla en la empresa, la resistencia al cambio y el traumatismo del cambio son elementos a tener en consideración. Por lo que al no existir un estándar en la iconografía de los modelos y de las herramientas a ocupar dentro de los programas genera una distorsión bastante mayor de lo que uno quisiera al momento de decidir cual herramienta comprar.

La idea de hacer una herramienta que pueda integrar varios servicios Web puede ser un elemento ambicioso desde el punto de vista de la carga histórica que esto significa y pensar que una herramienta va a ser capaz de hacerlo es esperar demasiado de ella por lo mismo. Ahora bien podríamos pensar en un programa capaz de integrar los servicios Web desarrollados desde cierto momento en adelante, pero eso también es muy difícil porque, si bien existen estándares para desarrollar elementos Web, procesos de negocios, esquemas XML y demases, esos estándares son débiles al momento en que los desarrolladores necesitan realizar o poner en marcha algún proyecto. Cabe preguntarse porque pasa esto, y quizás no se debe a una sola cosa sino a un conjunto de cosas como son los siguientes hechos:

i. Siempre se busca la simplicidad de las cosas y en general los estándares no entregan simplicidad en el ámbito del Web sino que para poder aumentar su potencial la complejidad ha aumentado, por ejemplo un formulario Web en un principio sólo poseía un código HTML hoy en día para que cumpla con todos los requerimientos debe tener un código WSDL, uno XML y además uno HTML. Si bien las herramientas generan todos estos códigos por si solas, significa que al momento de querer realizar un cambio y no poseer la herramienta correcta dicho cambio es muy complejo de realizar.

ii. El costo de poseer una herramienta si bien puede no ser alto, ya que hoy en día muchas de dichas herramientas son gratuitas, el tener alguien capacitado para manejarla puede ser muy caro, además de ser dependiente por parte de la empresa hacia la persona que maneja la herramienta por las mismas razones de estandarización antes explicadas.

iii. Dependencia entre la empresa que ocupa una determinada herramienta y la que da el soporte de dicha herramienta, en lo que respecta a seguridad y elementos de actualización de la herramienta.

Estos elementos de dependencia y complejidad son los menos deseados por una empresa, ya que busca flexibilidad y facilidad al momento de querer decidir en lo que respecta a las herramientas utilizadas para su entorno de trabajo vía Web.

Ahora bien se ve que el mundo del desarrollo de estas herramientas poco a poco va convergiendo a un mundo común, actualmente al menos ya están ocupando un motor común como es java para realizar sus procesos y correr sus motores lo que apunta que en algún momento converjan a más elementos comunes tanto en la modelación como en la ejecución. Así mismo se espera que un usuario que maneje una herramienta cualquiera pueda cambiarse a otra sin sufrir un trauma mayor.

El mundo del software gratuito ha recibido un gran aporte ya que todas las herramientas evaluadas ofrecen integración con los elementos gratuitos, si bien dicha integración no es perfecta, el hecho de que exista y que se muestre una preocupación por parte de las empresas en desarrollar dichos elementos es un gran avance. Además la distribución gratuita de algunas de estas herramientas hace que el usuario trate de acercarse más a los “nuevos” protocolos del Web.

Finalmente podemos decir que las herramientas de SOA y específicamente en el ámbito de la Orquestación de Servicio Web, aún están en desarrollo y que aunque no cumplan completamente con lo esperado, son elementos hoy en día necesarios de adoptar para que el desarrollo y el crecimiento del Web, pueda lograr una interrelación realmente fuerte entre empresas, estados y personas, y que dicha interrelación sea ordenada, entendible y manejable por cualquier persona o cualquier WSO.

9. Bibliografía

[1] PELTZ, Chris. Web Service Orchestration and Choreography. A look at WSCI and BPEL4W; WSJ Feature.

[2] PELTZ, Chris. Web Services Orchestration. A review of emerging technologies, tools and standards. Hewelett Packard, Co. 2003.

[3] CURBERA, F., GOLAND, Y., KLEIN, J. Business Process Execution Language for Web Services.http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbizspec/html/bpel1-1.asp. [Consulta: Abril 15 del 2006].

[4] Momentum [en línea] 2003. http://www.serviceoriented.org/Web_service_orchestration.html. [Consulta: Abril 1 del 2006].

[5] Momentum [en línea] 2003. http://www.serviceoriented.org/orchestration_engine.html. [Consulta: Abril 1 del 2006].

[6] CAITUIRO-MONGE, Hillary., RODRIGUEZ-MARTINEZ,Manuel. Net Traveler: A Framework for Autonomic Web Services Collaboration, Orchestration and Choreography in E-Government Information Systems, 2004.

[7] MEDJAHED, Brahim., BOUGUETTAY, Athman., ELMAGARMID, Ahmed K. ComposingWeb services on the SemanticWeb; 23 de Septiembre del 2003.

[8] CHATAIN, Thomas., JARD, Jard. Models for the Supervision of Web Services Orchestration with Dynamic Changes.

[9] BOOTH, D., HAAS, H., MACCABE, F., NEWCOMER, E., CHAMPION, M., FERRIS, C., ORCHARD, D.Web Services Architecture, W3C Working Group Note (11 February 2004). http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ [Consulta : Mayo 2 del 2006].

[10] He, H., HAAS, H., ORCHARD, D.Web Services Architecture Usage Scenarios, W3C Working Group Note (11 February 2004). http://www.w3.org/TR/2004/NOTE-ws-arch-scenarios-20040211/. [Consulta: Mayo 2 del 2006]

[11] SLEEPER, B. The five missing pieces of SOA. (10 September 2004). http://www.infoworld.com/article/04/09/10/37FEWebservmiddle_1.html. [Consulta: Mayo 5 del 2006]

[12] WILKES, L. The Web Services Protocol Stack en CBDI Web Services Roadmap - Guiding the Transition to Web Service and SO. [en línea] http://roadmap.cbdiforum.com/reports/protocols/ [Consulta: 10 Mayo 2006]

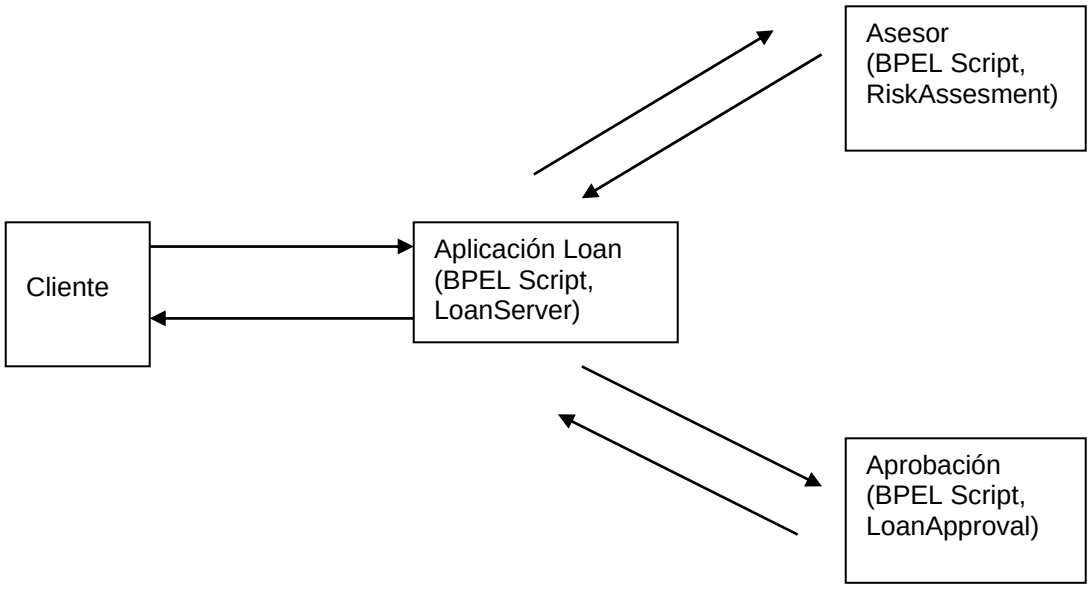
ANEXOS

A. BPEL Hello World

Ejemplo de código BPEL básico con el que fueron probadas en primera instancias las herramientas.

```
<?xml version="1.0" encoding="UTF-8"?>
<process
	xmlns="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
	xmlns:print="http://www.eclipse.org/tptp/choreography/2004/engine/Print"
	<!--Hello World - my first ever BPEL program -->
	<import importType="http://schemas.xmlsoap.org/wsdl/"
		location="../../test_bucket/service_libraries/tptp_EnginePrinterPort.wsdl"
		namespace="http://www.eclipse.org/tptp/choreography/2004/engine/Print" />
	<partnerLinks>
		<partnerLink  name="printService"
				partnerLinkType="print:printLink"
				partnerRole="printService"/>
	</partnerLinks>
	<variables>
		<variable	name="hello_world"
				messageType="print:PrintMessage" />
	</variables>
	<assign>
		<copy>
			<from><literal>Hello World</literal></from>
			<to>$hello_world.value</to>
		</copy>
	</assign>
	<invoke partnerLink="printService" operation="print" inputVariable="hello_world" />
</process>
```

B. Arquitectura del Servicio de ejemplo Loan



C. BPEL Loan Services

Uno de los archivo con el que fueron probado los sistema

```
<?xml version="1.0"?>
<bpws:process name="LoanService"
targetNamespace="http://www.mycompany.com/LoanService"
xmlns:tns="http://www.mycompany.com/LoanService"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:bpws="http://schemas.xmlsoap.org/ws/2003/03/business-process/"
xmlns:ns="http//demo.capeclear.com/LoanServices.wsdl"
xmls:ns1="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:ns2="http://loans.org/xsd/error-messages"
xmlns:ns3="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:ns4="http://schemas.xmlsoap.org/wsdl/"
xmlns:ns5="http://localhost.localdomain/LoanApproval.wsdl"
xmlns:ns6="http://localhost.localdomain/RiskAssessment.wsdl">
<bpws:import location="../LoanService.wsdl"
namespace="http://localhost.localdomain/LoanServices.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/">
</bpws:import>
<bpws:import location="../LoanApproval.wsdl"
namespace="http://localhost.localdomain/LoanApproval.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/">
</bpws:import>
<bpws:import location="../RiskAssessment.wsdl"
namespace="http://localhost.localdomain/RiskAssessment.wsdl"
importType="http://schemas.xmlsoap.org/wsdl/">
</bpws:import>
<bpws:partnerLinks>
<bpws:partnerLink myRole="loanService" name="customer"
partnerLinkType="ns:loanPartnerLinkType">
</bpws:partnerLink>
<bpws:partnerLink name="approver"
partnerLinkType="ns5:loanApprovalLinkType" partnerRole="approver">
</bpws:partnerLink>
<bpws:partnerLink name="assessor"
partnerLinkType="ns6:riskAssessmentLinkType"
partnerRole="assessor">
</bpws:partnerLink>
</bpws:partnerLinks>
<bpws:variables>
<bpws:variable name="creditInformationMessage"
messageType="ns:creditInformationMessage"/>
<bpws:variable name="approvalMessage"
messageType="ns:approvalMessage"/>
<bpws:variable messageType="ns:riskAssessmentMessage"
name="riskAssessmentMessage" />
```

```
</bpws:variables>
<bpws:faultHandlers>
<bpws:catch faultVariable="error" faultMessageType="ns:errorMessage">
<bpws:sequence>
<bpws:reply partnerLink="customer"
portType="ns:loanServicePT" operation="request"
variable="error"
faultName="ns:unableToHandleRequest">
</bpws:reply>
</bpws:sequence>
</bpws:catch>
</bpws:faultHandlers>
<bpws:eventHandlers></bpws:eventHandlers>
<bpws:sequence>
<bpws:receive partnerLink="customer" portType="ns:loanServicePT"
operation="request" variable="creditInformationMessage" createInstance="no">
</bpws:receive>
<bpws:invoke partnerLink="assessor"
portType="ns:riskAssessmentPT" operation="isLowRisk"
inputVariable="creditInformationMessage"
outputVariable="riskAssessmentMessage">
</bpws:invoke>
<bpws:switch>
<bpws:case condition="bpws:getVariableData('creditInformationMessage', 'amount') &gt;10000
or bpws:getVariableData('riskAssessmentMessage', 'level') !='low'">
<bpws:sequence>
<bpws:invoke partnerLink="approver"
portType="ns:loanApprovalPT"
operation="approve"
inputVariable="creditInformationMessage"
outputVariable="approvalMessage">
</bpws:invoke>
</bpws:sequence>
</bpws:case>
<bpws:otherwise>
<bpws:sequence>
<bpws:assign>
<bpws:copy>
<bpws:from
expression="'yes'"></bpws:from>
<bpws:to
variable="approvalMessage"
part="accept"/>
</bpws:copy>
</bpws:assign>
</bpws:sequence>
</bpws:otherwise>
</bpws:switch>
```

```
<bpws:reply partnerLink="customer" portType="ns:loanServicePT"
operation="request" variable="approvalMessage">
</bpws:reply>
</bpws:sequence>
</bpws:process>
```

D. LoanApproval.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="approval"
        targetNamespace="http://localhost.localdomain/LoanApproval.wsdl"
        xmlns:tns="http://localhost.localdomain/LoanApproval.wsdl"
        xmlns:lns="http://localhost.localdomain/LoanServices.wsdl"
        xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
        xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
        <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Based on the
WSDL file provided on the BPEL 1.1 specification</wsdl:documentation>
        <plnk:partnerLinkType name="loanApprovalLinkType">
                <plnk:role name="approver">
                        <plnk:portType name="lns:loanApprovalPT" />
                </plnk:role>
        </plnk:partnerLinkType>
        <wsdl:import location="LoanService.wsdl"
                namespace="http://localhost.localdomain/LoanServices.wsdl" />
        <wsdl:binding name="loanApprovalBinding"
                type="lns:loanApprovalPT">
                <soap:binding style="rpc"
                        transport="http://schemas.xmlsoap.org/soap/http" />
                <wsdl:operation name="approve">
                        <soap:operation
                                soapAction="capeconnect::loanApprovalPT#approve" />
                        <wsdl:input>
                                <soap:body use="literal" />
                        </wsdl:input>
                        <wsdl:output>
                                <soap:body use="literal" />
                        </wsdl:output>
                        <wsdl:fault name="loanProcessFault">
                                <soap:fault name="loanProcessFault" use="literal" />
                        </wsdl:fault>
                </wsdl:operation>
        </wsdl:binding>
        <wsdl:service name="ApprovalService">
                <wsdl:port binding="tns:loanApprovalBinding" name="approval">
                        <soap:address
                                location="http://localhost:8080/ccx/LoanApproval" />
                </wsdl:port>
        </wsdl:service>
</wsdl:definitions>
```

E. LoanService.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="LoanServices"
targetNamespace="http://localhost.localdomain/LoanServices.wsdl"
xmlns:tns="http://localhost.localdomain/LoanServices.wsdl"
xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ens="http://loans.org/xsd/error-
messages" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Based on the WSDL file
provided on the BPEL 1.1 specification</wsdl:documentation>
  <wsdl:types>
    <xsd:schema targetNamespace="http://loans.org/xsd/error-messages"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:ens="http://loans.org/xsd/error-
messages">
          <xsd:element name="errorCode" type="xsd:integer"/>
       </xsd:schema>
  </wsdl:types>
  <wsdl:message name="errorMessage">
    <wsdl:part name="errorCode" element="ens:errorCode"/>
  </wsdl:message>
  <wsdl:message name="creditInformationMessage">
    <wsdl:part name="firstName" type="xsd:string"/>
    <wsdl:part name="name" type="xsd:string"/>
    <wsdl:part name="amount" type="xsd:integer"/>
  </wsdl:message>
  <wsdl:message name="riskAssessmentMessage">
    <wsdl:part name="level" type="xsd:string"/>
  </wsdl:message>
  <wsdl:message name="approvalMessage">
    <wsdl:part name="accept" type="xsd:string"/>
  </wsdl:message>
  <wsdl:portType name="riskAssessmentPT">
    <wsdl:operation name="isLowRisk">
      <wsdl:input message="tns:creditInformationMessage"/>
      <wsdl:output message="tns:riskAssessmentMessage"/>
      <wsdl:fault name="loanProcessFault" message="tns:errorMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="loanApprovalPT">
    <wsdl:operation name="approve">
      <wsdl:input message="tns:creditInformationMessage"/>
      <wsdl:output message="tns:approvalMessage"/>
      <wsdl:fault name="loanProcessFault" message="tns:errorMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:portType name="loanServicePT">
```

```
    <wsdl:operation name="request">
      <wsdl:input message="tns:creditInformationMessage"/>
      <wsdl:output message="tns:approvalMessage"/>
      <wsdl:fault name="unableToHandleRequest" message="tns:errorMessage"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="loanServiceBinding" type="tns:loanServicePT">
    <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="request">
      <soap:operation soapAction="capeconnect::loanServicePT#request"/>
      <wsdl:input>
        <soap:body use="literal"/>
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal"/>
      </wsdl:output>
      <wsdl:fault name="unableToHandleRequest">
        <soap:fault name="unableToHandleRequest" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="LoanService">
    <wsdl:port name="service" binding="tns:loanServiceBinding">
      <soap:address location="http://localhost:8080/ccx/LoanService"/>
    </wsdl:port>
  </wsdl:service>
    <plnk:partnerLinkType xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
name="loanPartnerLinkType">
       <plnk:role name="loanService">
          <plnk:portType name="tns:loanServicePT"/>
       </plnk:role>
    </plnk:partnerLinkType>
</wsdl:definitions>
```

F. RiskAssessment.wsdl

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
   name="RiskAssessment"
   targetNamespace="http://localhost.localdomain/RiskAssessment.wsdl"
   xmlns:tns="http://localhost.localdomain/RiskAssessment.wsdl"
   xmlns:lns="http://localhost.localdomain/LoanServices.wsdl"
   xmlns:plnk="http://schemas.xmlsoap.org/ws/2003/05/partner-link/"
   xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
   xmlns:xsd="http://www.w3.org/2001/XMLSchema">
   <wsdl:documentation xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">Based on the WSDL
file provided on the BPEL 1.1 specification</wsdl:documentation>
   <plnk:partnerLinkType name="riskAssessmentLinkType">
      <plnk:role name="assessor">
         <plnk:portType name="lns:riskAssessmentPT"/>
      </plnk:role>
   </plnk:partnerLinkType>
   <wsdl:import
      location="LoanService.wsdl"
      namespace="http://localhost.localdomain/LoanServices.wsdl"/>
   <wsdl:binding name="riskAssessmentBinding" type="lns:riskAssessmentPT">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <wsdl:operation name="isLowRisk">
         <soap:operation
            soapAction="capeconnect:LoanServices:riskAssessmentPT#isLowRisk"/>
         <wsdl:input>
            <soap:body use="literal"/>
         </wsdl:input>
         <wsdl:output>
            <soap:body use="literal"/>
         </wsdl:output>
         <wsdl:fault name="loanProcessFault">
            <soap:fault name="loanProcessFault" use="literal"/>
         </wsdl:fault>
      </wsdl:operation>
   </wsdl:binding>
   <wsdl:service name="RiskAssessmentService">
      <wsdl:port binding="tns:riskAssessmentBinding" name="risk">
         <soap:address location="http://localhost:8080/ccx/RiskAssessment"/>
      </wsdl:port>
   </wsdl:service>
</wsdl:definitions>
```