Universidad de las Ciencias Informáticas

Facultad 9

Trabajo de Diploma para optar por el título de Ingeniero en Ciencias Informáticas.

Título: Propuesta de una estrategia para integrar y publicar componentes de software a través de servicios.

Autor: Isnán Rodríguez Macías

Tutor: Ing. Daniel Sampedro Bello

Ciudad de La Habana, mayo 2010

"Año 52 de la Revolución"

Declaración de autoría

Declaro que soy el único autor de este trabajo y autorizo a La Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

_____ _____

Isnán Rodríguez Macías Ing. Daniel Sampedro Bello

_____ _____/___/_____

Firma Fecha

Resumen

Ante la necesidad de integrar y publicar las funcionalidades de los componentes de software que se desarrollan en el polo productivo PetroSoft de la Facultad 9 ubicado en la Universidad de las Ciencias Informáticas, surge la idea de crear una infraestructura informática que dé solución a esta problemática.

El presente trabajo de diploma explora campos informáticos donde el tema fundamental son los referentes a técnicas de integración, tecnologías de integración y arquitecturas de software orientadas a servicios. Con la presente investigación se obtuvo la propuesta de una estrategia para publicar e integrar componentes de software. Con este fin se utilizó SOA como arquitectura para la construcción del sistema y se utilizó el patrón Entity Aggregation para la solución en cuanto a integración. Se utilizan además estándares y protocolos aprobados a nivel mundial relacionados con la arquitectura utilizada. Esta estrategia permitirá al polo construir software de alta calidad reutilizando los componentes que ya tiene desarrollados ahorrando tiempo, esfuerzo y recursos.

Palabras claves: Integración de Componentes de Software, SOA

Índice

Índice de Figuras

Índice de Tablas

Introducción

La Informática y las Telecomunicaciones forman cada día que pasa parte fundamental del desarrollo económico de cualquier país. El desarrollo acelerado que experimentan en la actualidad y la demanda creciente de software está permitiendo una evolución en el desarrollo de software donde la construcción de sistemas abiertos y distribuidos constituye la principal idea. Esto ha permitido que los paradigmas tradicionales de programación no solucionen de manera natural la complejidad requerida en estos tipos de sistemas. Por tanto, han surgido nuevos paradigmas como la programación orientada a componentes y la programación orientada a servicios que han revolucionado los procesos de construcción de soluciones informáticas.

En la actualidad se trabaja a nivel mundial siguiendo un enfoque conocido como: “Desarrollo de Software Basado en Componentes (DSBC)” donde la idea fundamental es desarrollar aplicaciones siguiendo un diseño que utilice componentes de software previamente desarrollados y reutilizables. El DSBC reutiliza piezas de código pre-elaboradas implicando diversos beneficios como las mejoras a la calidad, la reducción del ciclo de desarrollo y el mayor retorno sobre la inversión (ROI).

De esta manera, surgen técnicas de desarrollo de software que han ido perfeccionándose y madurando con el tiempo, desde la programación estructurada hasta la programación orientada a objetos. La programación orientada a componentes no constituye la evolución de la programación orientada a objetos sino que la utiliza contribuyendo significativamente en la reducción de los costos y acrecentando la reutilización. El DSBC no ha permitido obtener componentes de software de varios proveedores e integrarlos para conformar aplicaciones más complejas, objetivo que es buscado por la industria del software.

Persiguiendo este objetivo y tratando de satisfacer la necesidad de estandarizar los componentes ofreciendo interfaces abiertas pueden verse los tres esfuerzos que más han sobresalido: Microsoft ha puesto en el mercado las tecnologías COM, DCOM y COM+, la compañía Sun Microsystems (hoy perteneciente a la Corporación Oracle) ha mostrado su tecnología “Java Bean” y por último el consorcio Object Management Group ha desarrollado la tecnología CORBA.

En estos últimos años el país experimenta un desarrollo acelerado en la esfera informática, demostrado por la creación de nuevos centros y planes de estudio como es el caso de la Universidad de las Ciencias Informáticas (UCI), los Institutos Politécnicos de Informática (IPI) y los Joven Club de Computación y Electrónica. La informatización de la sociedad, la producción y exportación de software son objetivos importantes que han respaldado esta iniciativa aportando significativamente a la economía.

La UCI tiene organizada la producción en polos productivos. El polo PetroSoft ubicado dentro de esta disposición tenía una estructura enfocada a proyectos y semejante a la disposición del cliente. En otras palabras cuando un cliente solicitaba el desarrollo de cualquier aplicación de software, el polo se emprendía en la tarea de construirlo partiendo desde cero y teniendo como finalidad la entrega de un software de alta calidad y que cumpliera con las expectativas del cliente. Esto implicaba duplicar esfuerzos en cada proyecto. Un análisis profundo que originó una reestructuración del polo, posibilitó que ahora se enfoque en detectar las funcionalidades generales detectadas en los clientes. De esta forma, se crean componentes reutilizables que encapsulan estas funcionalidades y que pueden ser utilizados de un proyecto a otro siguiendo la orientación a un desarrollo de software basado en componentes y logrando una mejor infraestructura de producción de software.

El polo PetroSoft desarrolla componentes sobre diferentes plataformas y tecnologías ejemplo de ello se tiene el “Componente evaluador de expresiones matemáticas” que se desarrolla sobre C++ y el componente “Graficador de indicadores para el área de perforación de la industria petrolera” que se desarrolla sobre JavaScript. Razón por la cual muchas de las funcionalidades que brindan estos componentes tienen que ser implementadas en varias ocasiones; resultando tarea difícil utilizar las funcionalidades ya presentes en algún otro componente que no esté desarrollado en la misma plataforma y tecnología. Otro problema presente está relacionado con el soporte técnico y actualización a las aplicaciones realizadas debido a que es necesario ir directamente a donde se encuentra funcionando para poder realizar esta tarea. De la situación antes explicada se derivó el **problema a resolver**, constituido por la siguiente interrogante: ¿Cómo publicar e integrar los componentes de software producidos en los proyectos productivos en el polo PetroSoft?

Por el problema previamente expuesto y por las necesidades que tiene el polo en obtener una solución, se decidió cumplir con el siguiente **objetivo general:** Diseñar una estrategia para la integración y publicación

de los componentes de software producidos en el polo PetroSoft utilizando servicios. Como complemento para alcanzar el objetivo antes mencionado se han definido los siguientes **objetivos específicos**:

1. Evaluar las posibles tecnologías y herramientas de integración de componentes de software existentes.
2. Resumir los aspectos positivos de las tecnologías y herramientas evaluadas.
3. Diseñar una estrategia de integración de componentes de software para el polo productivo PetroSoft empleando servicios.

Como resultado de las características anteriormente expuestas, se enmarcó el **objeto de estudio** en el desarrollo e integración de componentes de software.

En derivación a la problemática en cuestión el **campo de acción** fue la integración de componentes de software en el polo PetroSoft.

Luego de examinar con profundidad las crecientes necesidades que aparecían en el polo como resultado de restructuraciones aplicadas, para lograr índices productivos mayores se llegó a la conclusión de que si se lograban identificar las tareas a realizar y las técnicas a aplicar para la integración de componentes de software en el polo a través de servicios, se conseguiría diseñar una estrategia de integración y publicación de dichos componentes para la construcción de soluciones informáticas de alta escalabilidad y prestaciones en el polo productivo PetroSoft que constituye la **hipótesis** de esta investigación.

Con la finalidad de resolver la problemática anteriormente planteada, se programó un conjunto de tareas investigativas y orientaciones específicas que establecieron el camino por donde se condujo para darle solución a los problemas mencionados, en función de las necesidades prácticas y cognoscitivas existentes. Las mismas son:

1. Describir el estado del arte relacionado con la integración y publicación de componentes de software a través de servicios.
2. Identificar las metodologías de desarrollo de software existentes relacionadas con el ensamblaje de componentes de software.

3. Evaluar las metodologías de desarrollo de software existentes relacionadas con el ensamblaje de componentes de software.
4. Identificar las herramientas y tecnologías de software, que faciliten la integración de componentes de software.
5. Evaluar las herramientas y tecnologías de software, que faciliten la integración de componentes de software.
6. Generalizar los elementos identificados para obtener la propuesta de integración de componentes.
7. Describir la estrategia de integración.
8. Probar la estrategia de integración y publicación de componentes.

Para cumplir con los objetivos trazados en la presente investigación se emplearon los métodos científicos teóricos: histórico lógico, el analítico sintético y el causal y dentro de los métodos empíricos: la entrevista.

El método histórico lógico: permitió la comprensión del derrotero histórico de tecnologías, herramientas y metodologías relacionas con los componentes de software y las Arquitecturas Orientadas a Servicios (SOA).

El método analítico sintético: permitió a medida que se desarrolló la investigación ir cuantificando aspectos esenciales para descripción y comprensión de la estrategia de integración y publicación de componentes de software.

El método causal permitió estudiar los factores que provocan la necesidad de una estrategia para la integración y publicación de componentes de software en el Polo Productivo PetroSoft.

También fue útil la entrevista para obtener de un conjunto de expertos los criterios necesarios para el empleo de las técnicas propuestas en el Método Delphi para la evaluación de la propuesta.

Cuando hayan sido cumplidas completamente todas las tareas y las investigaciones necesarias se espera alcanzar entre los posibles resultados, la identificación de los métodos que favorecerán plantear una estrategia que permita publicar e integrar componentes de software en el polo. Para plantear esta estrategia es imprescindible realizar una intensa investigación de los campos relacionados con el DSBC y métodos de integración de software obteniendo de esta forma beneficios concretos.

En el Capítulo 1 se realiza el estudio del estado del arte y la fundamentación teórica del tema que compete. Se muestran los conceptos más utilizados a nivel mundial de DSBC y SOA. Se hace un recorrido por la evolución histórica de estos.

En el Capítulo 2 se propone una estrategia para darle solución a la problemática existente en el polo de cómo integrar y publicar los componentes de software. Se utiliza para la solución un enfoque basado en SOA y se relacionan todos los aspectos tecnológicos necesarios para su puesta en práctica.

En el Capítulo 3 se utiliza el método de Encuesta para evaluar la solución propuesta. En este proceso de evaluación intervienen diferentes expertos en los temas de metodologías de desarrollo de software, procesos de integración de software y arquitectura de software. Obtenido de esta manera un consenso sobre los pro y los contra introducidos con la aplicación de la estrategia propuesta.

Capítulo I: Fundamentación teórica

1.1 Introducción

EL DSBC es una idea que ha sido puesta en práctica por las empresas desarrolladoras de software considerando los beneficios que trae su adopción. Aunque muchos han sido los esfuerzos para estandarizar la producción de estos componentes, aún no existe un consenso común en cómo construir estos componentes, de manera que puedan ser utilizados por terceras partes independientemente de su diseño, plataforma y lenguaje de desarrollo. Ante la necesidad de utilizar varios componentes desarrollados por terceros han surgido diferentes métodos de integración.

En el presente capítulo se describen las principales características relacionadas con los componentes de software, también se describen las tecnologías a emplear para el diseño de una infraestructura capaz de exponer las funcionalidades de los componentes desarrollados en el polo de forma estándar, abstrayendo a los usuarios de su implementación y haciéndolos independientes de la plataforma y del lenguaje de programación a la hora de utilizarlos.

1.2 Estado del arte

A nivel internacional existen muchas soluciones de integración, cada una se adapta a las necesidades propias del entorno para la cual fueron concebidas. En unos casos el proceso de integración se centra en cómo integrar componentes construidos por terceras partes como Java Beans, COM y CORBA con las aplicaciones que se encuentran en proceso de desarrollo. En otros casos la integración se utiliza para lograr interoperabilidad entre sistemas que ya se encuentran funcionando pero no se estableció un mecanismo estándar cuando se construyeron para el intercambio de información con las mismas. Los casos mencionados anteriormente son los más relevantes en lo que respecta a integración pero existen muchos más ejemplos relacionados con el tema.

Para la solución de integración entre sistemas heterogéneos ha aparecido también la tecnología middleware que introduce una capa entre las aplicaciones, siendo esta capa la que se encarga de actuar como mediador entre estas aplicaciones. El middleware es físicamente una aplicación que se encarga de

la comunicación permitiéndonos abstraernos de la complejidad y heterogeneidad de las redes de comunicación, sistemas operativos y lenguajes de programación.

Con el surgimiento de SOA los problemas de integración comienzan a solucionarse de una manera más sencilla y eficiente. SOA aporta la solución más práctica para aplicar, logrando estandarizar la comunicación. La orientación a servicios ofrecida por esta arquitectura es tanto un marco de trabajo para el desarrollo de software como un marco de trabajo de implementación. Es importante señalar que para integrar es necesario tener de antemano algo que valga la pena integrar, tampoco es utilizar SOA por utilizarlo si no hay que tener bien claro que SOA representa la mejor opción a emplear.

La UCI posee numerosos proyectos en los cuales se han tenido que aplicar diferentes técnicas y tecnologías de integración para solucionar los problemas que se han presentado de heterogeneidad entre aplicaciones o componentes. En cada problema presente se ha empleado una manera distinta para solucionarlo, por tanto, no existe un patrón para las soluciones de integración debido a que cada problema requiere un modelo diferente de concepción de la integración.

Actualmente los proyectos trabajan en el desarrollo de componentes reutilizables, que luego son expuestos en los repositorios de componentes a donde se puede acceder y descargar estos componentes para utilizarlos en las soluciones informáticas que se encuentran en desarrollo. Entre los esfuerzos que algunos proyectos trabajan para crear soluciones estándar de integración y publicación de componentes se encuentra la de crear aplicaciones que expongan las funcionalidades de los componentes como servicios. Concretamente existe en el proyecto ERP el desarrollo de una aplicación que será capaz de consumir componentes PHP y exponer sus funcionalidades como servicios, pero aún no existe ningún proyecto que trabaje en crear toda una infraestructura para consolidar esta idea de integración y publicación.

1.3 Componentes de Software

La tendencia a desarrollar software mediante componentes que han sido elaborados por separados no es una idea nueva; muchos son los autores que plantean que simplemente es la evolución de la metodología orientada a objetos. Y es que efectivamente muchas de las características de los componentes para el desarrollo parten de la idea del diseño orientado a objetos.

El desarrollo de software basado en componentes tiene su historia. Uno de los frutos de la revolución industrial fue el desarrollo por componentes, surgido a partir de la necesidad de estandarizar los elementos de los productos realizados en línea, como por ejemplo los automóviles, este proceso de desarrollo basado en componentes trae consigo la aparición de conceptos nuevos tales como estandarización, adaptación y ensamblaje de componentes. De todos estos principios se nutrieron los desarrolladores de software para empezar a idear una estrategia similar que le brindara los beneficios que proporcionó en otras áreas.

Diferentes autores internacionalmente plantean definiciones para componente de software entre las que se encuentran:

- Un componente es en esencia una pieza de código pre elaborado que encapsula alguna funcionalidad expuesta a través de interfaces estándar. Los componentes son los "ingredientes de las aplicaciones", que se juntan y combinan para llevar a cabo una tarea. Es algo muy similar a lo que se puede observar en el equipo de música que se tiene en la sala. Cada componente de aquel aparato ha sido diseñado para acoplarse perfectamente con sus pares, las conexiones son estándar y el protocolo de comunicación está ya preestablecido. Al unirse las partes, obtenemos música para nuestros oídos. (Casal Terreros, 2010)
- Un componente es una unidad binaria de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio. (Alden Szyperski, 2010)
- Una implementación que (a) realiza un conjunto de funciones relacionadas, (b) puede ser independientemente desarrollado, entregado e instalado, (c) tiene un conjunto de interfaces para los servicios proporcionados y otro para los servicios requeridos, (d) permite tener acceso a los datos y al comportamiento sólo a través de sus interfaces, (e) opcionalmente admite una configuración controlada. (IBM-WebSphere01, 2010)
- Un componente de software es un fragmento de un sistema software que puede ser ensamblado con otros fragmentos para formar piezas más grandes o aplicaciones completas. (Software Engineering Institute Carnegie Mellon, 2010)

Como conclusión de estas definiciones se puede hacer las siguientes valoraciones: en las definiciones ofrecidas internacionalmente para componentes de software, un concepto no está fuera del ámbito del otro sino que entre ellos se complementan y construyen, mostrando a los componentes de software como partes del software que se pueden combinar para confeccionar un conjunto mayor como un subsistema, un sistema o incluso otro componente. Un componente juega el papel de una unidad de software reutilizable que puede interoperar con otros módulos del software mediante sus interfaces. Un componente define una o más interfaces desde donde se puede tener acceso a los servicios que este ofrece.

Los componentes presentan además grupo un de características indispensables que son necesarias cuando se quiere catalogar un producto como un componente de software o no. A continuación, son relacionadas estas características, descritas por Maribel Ariza Rojas y Juan Carlos Molina García, en su artículo: ”Introducción y principios básicos del desarrollo de software basado en componentes”: (Molina García, y otros, 2004)

- **Identificable**: Debe tener una identificación que permita acceder fácilmente a sus servicios y que permita su clasificación.
- **Auto contenido**: Un componente no debe requerir de la utilización de otros para finiquitar la función para la cual fue diseñado.
- **Puede ser remplazado por otro componente**: Se puede reemplazar por nuevas versiones u otro componente que lo reemplace y mejore.
- **Con acceso solamente a través de su interfaz:** Debe asegurar que estas no cambiarán a lo largo de su implementación.
- **Sus servicios no varían**: Las funcionalidades ofrecidas en su interfaz no deben variar, pero su implementación sí.
- **Bien Documentado**: Un componente debe estar correctamente documentado para facilitar su búsqueda si se quiere actualizar, integrar con otros, adaptarlo, etc.
- **Es genérico:** Sus servicios deben servir para varias aplicaciones.
- **Reutilizado dinámicamente:** Puede ser cargado en tiempo de ejecución en una aplicación.
- **Independiente de la plataforma:** Hardware, Software, S.O.

Al realizar un estudio de las definiciones anteriormente planteadas se puede llegar a algunas conclusiones: los componentes son fracciones de software que se integran con otros componentes para formar otro componente, un sistema o subsistema. Un componente de software juega un rol fundamental como parte de software reutilizable que puede integrarse con otros módulos de software mediante sus interfaces. Un componente tiene definida una o más interfaces que permiten el acceso a los servicios que este componente ofrece.

1.3.1 Tipos de componentes de software

La figura 1 muestra los componentes de software que podemos encontrar y sus relaciones con los demás componentes:

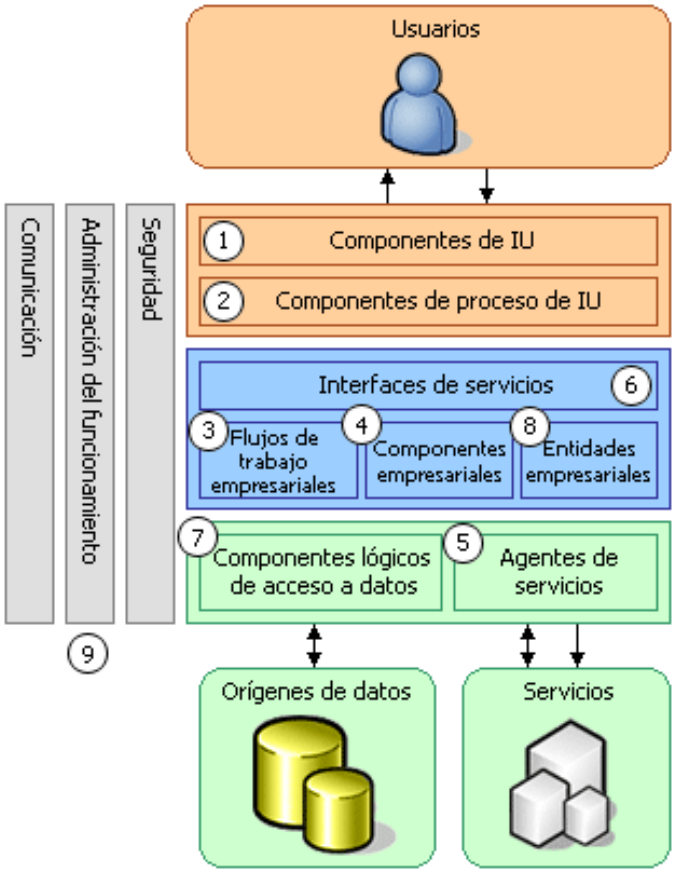

Figura 1: Tipos de Componentes y sus relaciones. (Microsoft Corporation, 2006)

Componentes de software: (Microsoft Corporation, 2006)

1. **Componentes de interfaz de usuario (IU).** La mayor parte de las soluciones necesitan ofrecer al usuario un modo de interactuar con la aplicación. En el ejemplo de aplicación comercial, un sitio Web permite al cliente ver productos y realizar pedidos, y una aplicación basada en el entorno operativo Microsoft Windows permite a los representantes de ventas escribir los datos de los pedidos de los clientes que han telefoneado a la empresa. Las interfaces de usuario se implementan utilizando formularios de Windows Forms, páginas Microsoft ASP.NET, controles u otro tipo de tecnología que permita procesar y dar formato a los datos de los usuarios, así como adquirir y validar los datos entrantes procedentes de éstos.
2. **Componentes de proceso de usuario.** En un gran número de casos, la interactuación del usuario con el sistema se realiza de acuerdo a un proceso predecible. Por ejemplo, en la aplicación comercial, podríamos implementar un procedimiento que permita ver los datos del producto. De este modo, el usuario puede seleccionar una categoría de una lista de categorías de productos disponibles y, a continuación, elegir uno de los productos de la categoría seleccionada para ver los detalles correspondientes. Del mismo modo, cuando el usuario realiza una compra, la interactuación sigue un proceso predecible de recolección de datos por parte del usuario, por el cual éste en primer lugar proporciona los detalles de los productos que desea adquirir, a continuación los detalles de pago y, por último, la información para el envío. Para facilitar la sincronización y organización de las interactuaciones con el usuario, resulta útil utilizar componentes de proceso de usuario individuales. De este modo, el flujo del proceso y la lógica de administración de estado no se incluye en el código de los elementos de la interfaz de usuario, por lo que varias interfaces podrán utilizar el mismo "motor" de interactuación básica.
3. **Flujos de trabajo empresariales**. Una vez que el proceso de usuario ha recopilado los datos necesarios, éstos se pueden utilizar para realizar un proceso empresarial. Por ejemplo, tras enviar los detalles del producto, el pago y el envío a la aplicación comercial, puede comenzar el proceso de cobro del pago y preparación del envío. Gran parte de los procesos empresariales conllevan la realización de varios pasos, los cuales se deben organizar y llevar a cabo en un orden determinado. Por ejemplo, el sistema empresarial necesita calcular el valor total del pedido, validar la información de la tarjeta de crédito, procesar el pago de la misma y preparar el envío del

producto. El tiempo que este proceso puede tardar en completarse es indeterminado, por lo que sería preciso administrar las tareas necesarias, así como los datos requeridos para llevarlas a cabo. Los flujos de trabajo empresariales definen y coordinan los procesos empresariales de varios pasos de ejecución larga y se pueden implementar utilizando herramientas de administración de procesos empresariales, como BizTalk Server Orchestration.

4. **Componentes empresariales**. Independientemente de si el proceso empresarial consta de un único paso o de un flujo de trabajo organizado, la aplicación requerirá probablemente el uso de componentes que implementen reglas empresariales y realicen tareas empresariales. Por ejemplo, en la aplicación comercial, deberá implementar una funcionalidad que calcule el precio total del pedido y agregue el costo adicional correspondiente por el envío del mismo. Los componentes empresariales implementan la lógica empresarial de la aplicación.
5. **Agentes de servicios**. Cuando un componente empresarial requiere el uso de la funcionalidad proporcionada por un servicio externo, tal vez sea necesario hacer uso de código para administrar la semántica de la comunicación con dicho servicio. Por ejemplo, los componentes empresariales de la aplicación comercial descrita anteriormente podría utilizar un agente de servicios para administrar la comunicación con el servicio de autorización de tarjetas de crédito y utilizar un segundo agente de servicios para controlar las conversaciones con el servicio de mensajería. Los agentes de servicios permiten aislar las idiosincrasias de las llamadas a varios servicios desde la aplicación y pueden proporcionar servicios adicionales, como la asignación básica del formato de los datos que expone el servicio al formato que requiere la aplicación.
6. **Interfaces de servicios**. Para exponer lógica empresarial como un servicio, es necesario crear interfaces de servicios que admitan los contratos de comunicación (comunicación basada en mensajes, formatos, protocolos, seguridad y excepciones, entre otros) que requieren los clientes. Por ejemplo, el servicio de autorización de tarjetas de crédito debe exponer una interfaz de servicios que describa la funcionalidad que ofrece el servicio, así como la semántica de comunicación requerida para llamar al mismo. Las interfaces de servicios también se denominan *fachadas empresariales*.
7. **Componentes lógicos de acceso a datos**. La mayoría de las aplicaciones y servicios necesitan obtener acceso a un almacén de datos en un momento determinado del proceso empresarial. Por ejemplo, la aplicación empresarial necesita recuperar los datos de los productos de una base de

datos para mostrar al usuario los detalles de los mismos, así como insertar dicha información en la base de datos cuando un usuario realiza un pedido. Por tanto, es razonable abstraer la lógica necesaria para obtener acceso a los datos en una capa independiente de componentes lógicos de acceso a datos, ya que de este modo se centraliza la funcionalidad de acceso a datos y se facilita la configuración y el mantenimiento de la misma.

8. **Componentes de entidad empresarial**. La mayoría de las aplicaciones requieren el paso de datos entre distintos componentes. Por ejemplo, en la aplicación comercial es necesario pasar una lista de productos de los componentes lógicos de acceso a datos a los componentes de la interfaz de usuario para que éste pueda visualizar dicha lista. Los datos se utilizan para representar entidades empresariales del mundo real, como productos o pedidos. Las entidades empresariales que se utilizan de forma interna en la aplicación suelen ser estructuras de datos, como conjuntos de datos, DataReader o secuencias de lenguaje de marcado extensible (XML), aunque también se pueden implementar utilizando clases orientadas a objetos personalizadas que representan entidades del mundo real necesarias para la aplicación, como productos o pedidos.
9. **Componentes de seguridad, administración operativa y comunicación.** La aplicación probablemente utilice también componentes para realizar la administración de excepciones, autorizar a los usuarios a que realicen tareas determinadas y comunicarse con otros servicios y aplicaciones.

1.3.2 Beneficios del desarrollo de software basado en componentes

Optar por la aplicación del paradigma de programación orientada a componentes donde la idea principal es solo escribir código para integrar estos componentes logrando que funcionen como un solo sistema; significa escalar un peldaño más en lo que constituye la evolución del DSBC. A continuación se ponen de manifiesto las principales ventajas ofrecidas por este paradigma según la visión de Julio Casal Terreros: (Casal Terreros, 2010)

- **Reutilización del software:** La posibilidad de utilizar constantemente componentes de software que se encuentren previamente desarrollados por nuestra entidad o por terceros, aporta la ventaja de reducir notablemente los tiempos de desarrollo y el esfuerzo del equipo de trabajo.

- **Simplifica las pruebas:** El uso de componentes de software permite que los mismos puedan ser probados de manera unitaria para garantizar que cumplen con las funcionalidades para las que fue diseñado. Esta característica influye directamente en la reducción de los márgenes de error así como en la ubicación de manera más acelerada de posibles fallas.
- **Simplifica el mantenimiento del sistema:** En un sistema desarrollado a partir de componentes en donde los mismos están débilmente acoplados entre ellos, los desarrolladores se encuentran en la libertad de agregar o separar componentes o cambiar los mismos para lograr cumplir con éxito con las funcionalidades del sistema.
- **Mayor calidad:** Debido a que los componentes pueden ser constantemente mejorados por los desarrolladores de los mismos, los sistemas ensamblados a partir de estos incrementarán su calidad con el paso del tiempo debido a que nuevas versiones de los componentes podrán ser adheridas o sustituirán viejos componentes agregándole nuevas funcionalidades, mejor concebidas e implementadas.
- **Ciclos de desarrollo más cortos:** En el DSBC, la adición de un nuevo componente de software, tomará días como mucho. En caso de que el equipo de desarrollo deba asumir la implementación de las funcionalidades dadas, podría tardarse meses o aún más tiempo en llevar la actividad a término e integrarla al sistema.
- **Mejor Retorno sobre la Inversión:** Utilizando de manera correcta la tecnología de componentes, podrá garantizarse la reintegración de toda la inversión realizada en la fase de aprovisionamiento, donde pueden haberse efectuado gastos concernientes a compras de componentes o licenciamiento de los mismos.
- **Funcionalidad mejorada:** Para usar un componente que contenga la implementación de un requisito dado, sólo se necesita entender su naturaleza, más no sus detalles internos. De esta manera, funcionalidades que anteriormente podrían resultar imprácticas para la entidad que se encuentre desarrollando el software, se vuelven ahora completamente asequibles.

1.4 Niveles de Integración de software

La integración constituye la habilidad de recoger elementos o aspectos de una entidad o de varias e incorporarlos al ente o conjunto de organismos. En la industria del software y específicamente en la arquitectura de software busca la relación entre el espacio interior con el espacio exterior. La arquitectura

de integración persigue una manera eficiente y flexible de combinar recursos y reutilizar soluciones con el objetivo de mejorar operaciones. Además, provee una vista única donde se consolidan conectores que definen y especifican el comportamiento e interacción entre elementos de software que abstraen el negocio (sistemas, subsistemas, y componentes). Constituye la conexión y la colaboración mutua entre actividades o procesos; lo que sugiere que la integración es más que una actividad simple de unir partes.

La integración de sistema de software resulta una tarea difícil de realizar en la que intervienen innumerables factores como plataformas, lenguajes de desarrollo, tecnologías, tiempo y costo entre otros indicadores. Dependiendo de ambiente para realizar la integración, los componentes a integrar, la preparación que tienen estos componentes para integrarse, complejidad de la integración y los procedimientos y criterios para la integración se pueden establecer cuatro niveles de integración de software en los cuales pueden estar involucrados estos sistemas a integrar.

1.4.1 Nivel 1: Integración Punto a Punto

La solución es relativamente simple e ideal cuando el número de componentes que tienen que intercambiar datos es muy reducido y estático. La comunicación se realiza entre las interfaces de cada uno de los componentes, con dominio rígido de los flujos de información, no se alcanza ninguna especialización funcional, ni la colaboración procesual de las áreas de proceso asistidas por sistemas de software. No se logra un modelado en el sistema de integración.

La siguiente figura muestra un ejemplo de tres componentes relacionados en este nivel de integración:

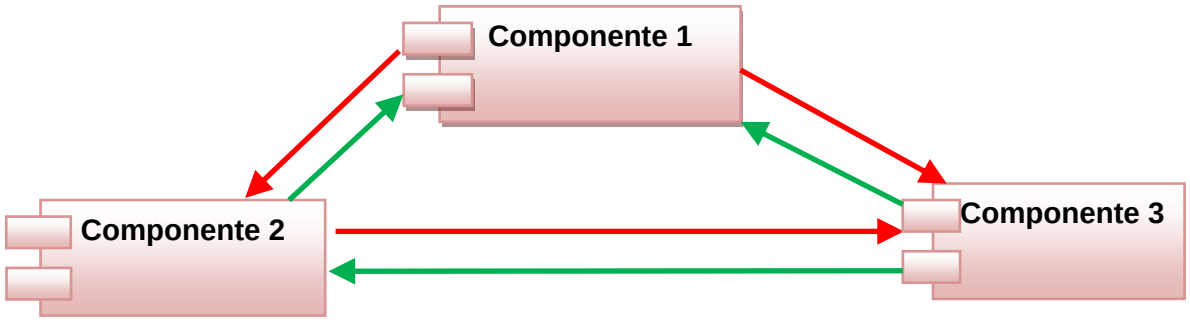

Figura 2: Integración punto a punto

1.4.2 Nivel 2: Integración Estructural

Está basada en tecnologías middleware, y con ella se logra estandarizar, gestionar y controlar el intercambio de información entre los componentes de software. En este nivel se tienen en cuenta dos aspectos: (1) estructuras centrales que controlan el intercambio de información (Orquestadores, proveedores de servicios, variables globales), (2) uso de tecnologías Middleware en las que se consolidan las reglas de negocio y las transacciones entre las aplicaciones. En este nivel las soluciones cuentan con interfaces integradas y fuentes de datos comunes, pero no se integran con componentes externos a su ambiente de infraestructura. La integración abstrae el sistema de software como un todo, obviando su entorno.

La siguiente figura 3 muestra un ejemplo de este nivel de integración:

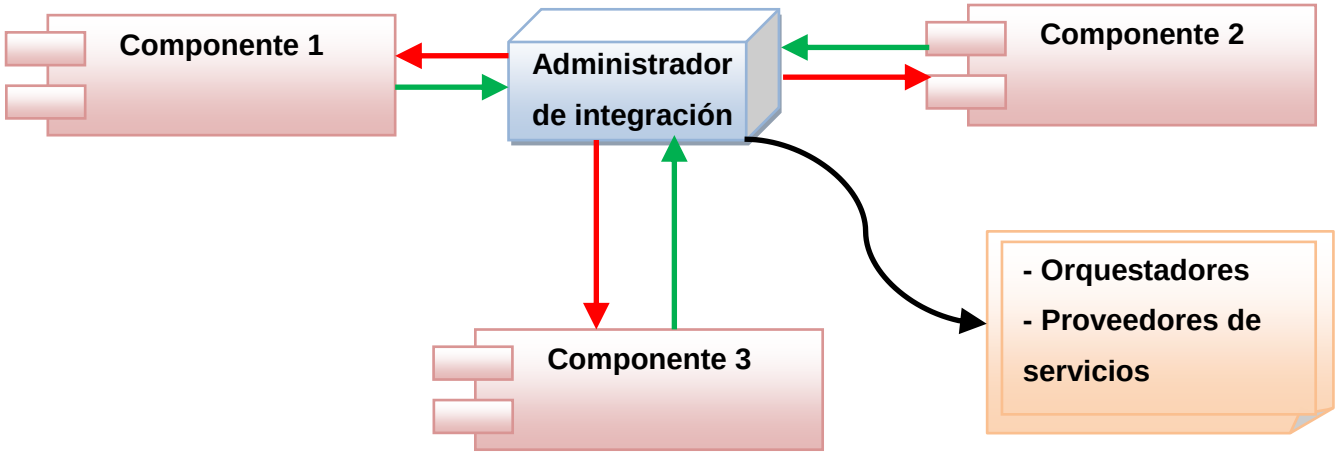

Figura 3: Integración estructural

1.4.3 Nivel 3: Integración de Procesos

Se presenta desde una arquitectura de aplicaciones que permite compartir información entre estas para manejar el flujo de información organizacional de los procesos. Se desarrolla un modelo de negocio común que cubre la totalidad de la empresa. Utiliza tecnología middleware sofisticada para poner en práctica el modelo de negocio en la capa de integración. No sólo se toman en cuenta las reglas del negocio, sino también el flujo de procesos, roles de procesos y estándares.

La figura 4 que se presenta a continuación ofrece una panorámica de este nivel de integración:

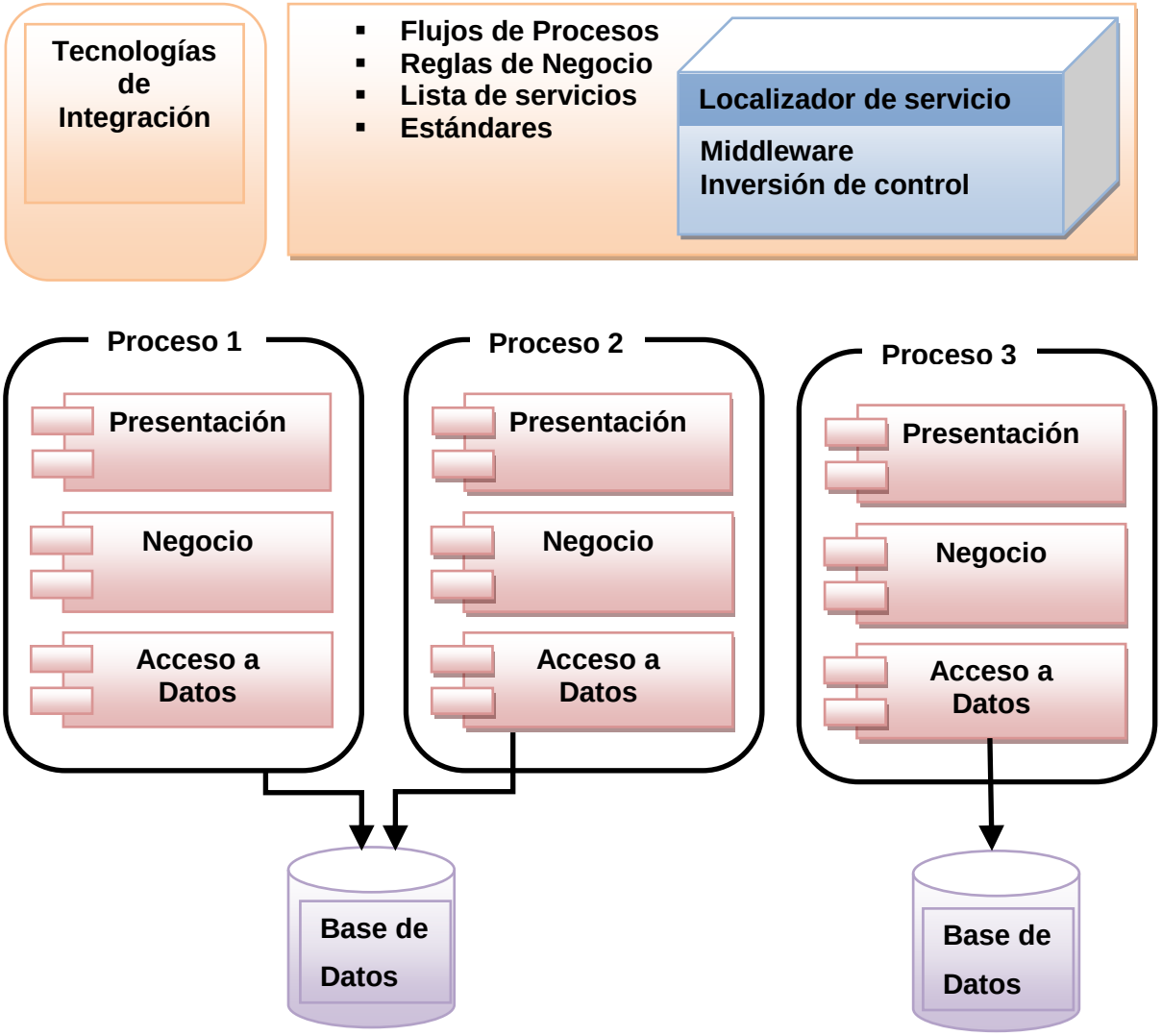

Figura 4: Integración de procesos

1.4.4 Nivel 4: Integración Externa

Se ve la solución desde un entorno empresarial interno y externo. La influencia tecnológica, la transformación de procesos de negocio y las nuevas estructuras, para redefinir la solución desde el punto de vista productor y consumidor. Estas soluciones se apoyan en tecnologías de Integración de Aplicaciones de Empresa (EAI) para transformar el negocio. En ocasiones se necesitan administradores

de integración que se conecten directamente a elementos clientes y proveedores en realizaciones de operaciones internas. Estas soluciones contemplan nuevas capacidades de adaptación a los cambios organizacionales internos y externos del entorno corporativo. La figura 5 presentada a continuación muestra la vista de lo que puede ser un ejemplo de este nivel:

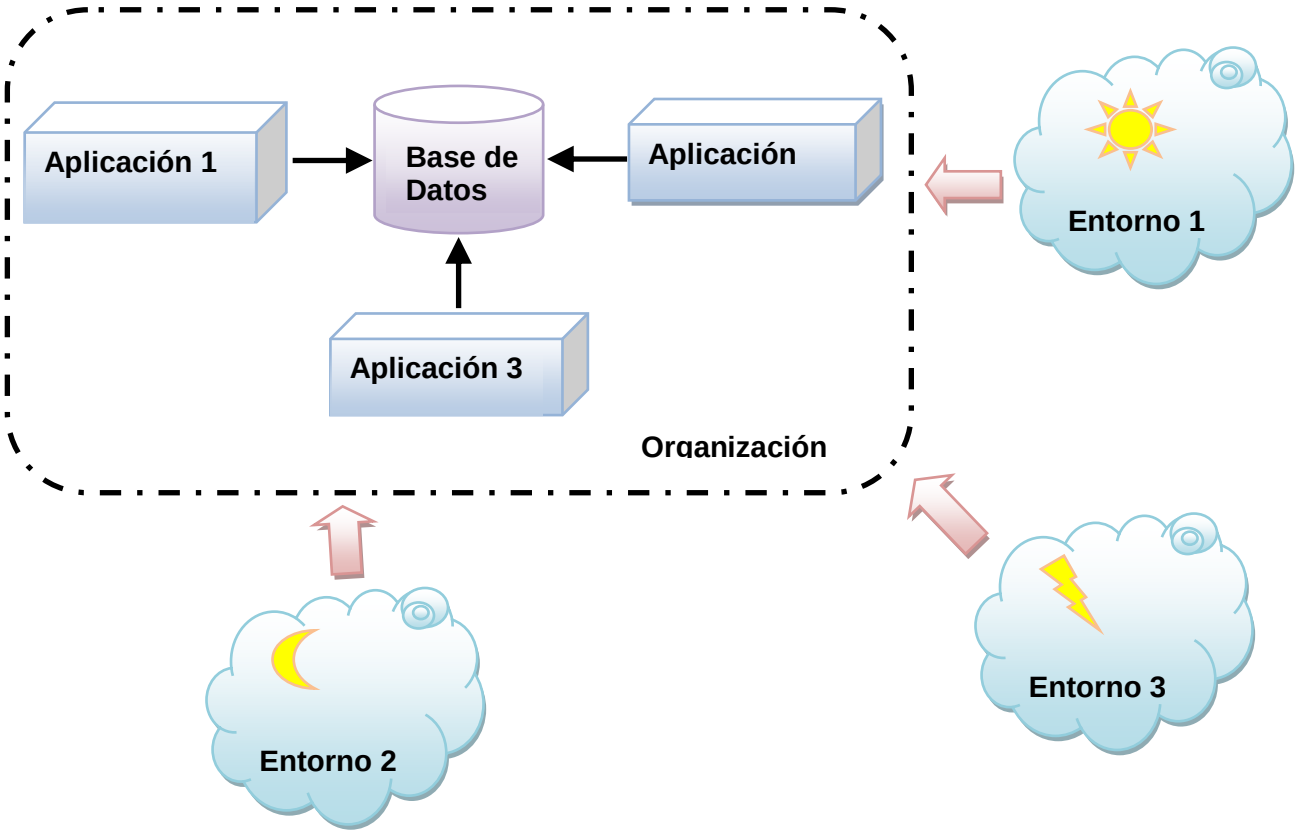

Figura 5: Integración externa

1.5 Patrones de Integración

Los patrones de integración en la informática son las soluciones a problemas comunes en el desarrollo de software, donde es necesario conectar sistemas aislados y heterogéneos dentro de una organización o entre varias y posibilitar su funcionamiento como un único sistema. A continuación se presentan los patrones más importantes en el ámbito de la integración de software.

1.5.1 Entity Aggregation

Este patrón introduce una capa de agregación de entidades la cual provee una conexión física con las mismas, hacia sus fuentes originales de datos, las cuales soportan el acceso y manipulación de las instancias de dichas entidades. La capa, *Entity Aggregation,* proporciona una representación lógica de las entidades a nivel empresarial con conexiones físicas que apoyan el acceso y la actualización en sus instancias respectivas en los repositorios. (Microsoft Corporation, 2010)

Este patrón presenta una vista general de toda la información recopilada en los repositorios al usuario. Los datos presentes en los repositorios son mostrados al usuario como se presenta en la figura 6:

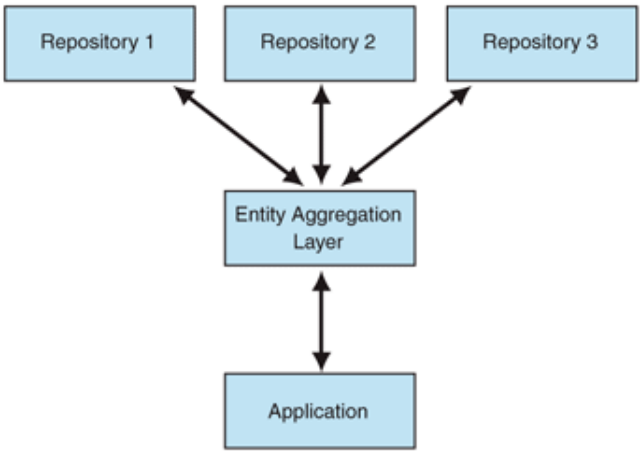

Figura 6: Ejemplo del patrón *Entity Aggregation* (Microsoft Corporation, 2010)

El establecimiento de *Entity Aggregation* implica un proceso de dos pasos:

1) Definición de la representación de toda la empresa que proporciona una representación coherente de la entidad unificada.
2) Implementación de una conexión física bidireccional entre esta representación y sus instancias respectivas en los repositorios de servicios.

1.5.1.1 Beneficios: (Microsoft Corporation, 2010)

- **Consenso.** Fuerza el consenso a través de las empresas y unidades funcionales sobre la manera en que las entidades están representadas en un nivel empresarial.
- **Vista única.** Permite una vista única de las entidades empresariales claves, como cliente, cuenta, producto.
- **Mejora del acceso a la información.** Un punto de vista de nivel empresarial de las entidades claves del negocio permite a las aplicaciones tener acceso inmediato a la información correspondiente a estas entidades. El acceso a la información no está limitado por los depósitos que los albergan.
- **Reducción de la disonancia semántica.** Elimina la disonancia semántica a través de las aplicaciones existentes que trabajan en los mismos elementos de datos desde múltiples repositorios.
- **Céntrico.** Es compatible con una ubicación central para la validación de los datos dentro de los repositorios.
- **Reducción del impacto del cambio.** Reduce el impacto potencial de los cambios en los repositorios. Dependiendo de la naturaleza de los cambios realizados, esta capa puede continuar sirviendo a las necesidades de las aplicaciones, mientras que estos cambios están en marcha.

1.5.1.2 Riesgos: (Microsoft Corporation, 2010)

- **Capa adicional.** Introduce una capa adicional de arquitectura que pueda afectar negativamente el rendimiento final de la solución.
- **Consenso.** Es necesario un consenso a través de las unidades de negocio en la definición de la escala de representación de las entidades empresariales.
- **Reingeniería de las aplicaciones.** Las aplicaciones existentes que están estrechamente unidas a una serie de repositorios tendrían que ser rediseñadas para dar cabida a la capa de la arquitectura nueva. Además, no siempre es posible rediseñar algunas soluciones, sobre todo para las aplicaciones empaquetadas.

Este patrón ofrece la mejor solución a emplear para garantizar el proceso de integración que se quiere lograr en el polo con los componentes de software. Además, ofrece ventajas sobre otros patrones de integración debido a que es fácil de implementar, ofrece una visión general de la información que se

encuentra en los repositorios donde la lógica y la complejidad quedan ubicadas en capa añadida por el patrón y no introduce cambios significativos en la infraestructura actual del polo.

1.5.2 Process Integration

Define un modelo de proceso de negocio que describe los pasos individuales que componen la función de negocios complejos. Crea un componente de proceso administrador distinto, que puede interpretar varias instancias simultáneas de este modelo y que puede interactuar con las aplicaciones existentes para llevar a cabo los distintos pasos del proceso. (Microsoft Corporation, 2010)

Este patrón permite una administración centralizada de los procesos que se integran independientemente del dominio de estos, debido a que solo interpreta las estructuras de estos procesos de forma básica. El polo necesita integrar componentes y no procesos por consiguiente no es la solución a emplear para solucionar los problemas de integración y publicación de los componentes.

1.5.3 Portal Integration

Es comparativamente un simple formulario de integración. Es popular porque es generalmente mucho más fácil de aplicar y menos intrusivo que otros tipos sofisticados de integración, como *Process Integration.* Con este patrón, el negocio de proceso describe la secuencia de tareas, no tienen que estar representados en un modelo de proceso exacto sino que reside en la cabeza del usuario. Un ejemplo del patrón Portal Integration es mostrado en la figura 8, interactúan las aplicaciones individuales mediante la integración de datos, integración funcional, o la integración de presentación. (Microsoft Corporation, 2010)

La integración dentro de este patrón puede presentarse como integración de datos, integración funcional o integración de presentación. Este patrón presenta cierta complejidad a la hora de ser implementado y no ofrece una solución eficiente a emplear en el polo debido a que representa principalmente un solución intermedia mientras se definen claramente los procesos de negocios, una vez definidos estos procesos se utiliza un modelo de procesos de negocios.

1.5.4 Message Broker

En este patrón la comunicación entre aplicaciones implica sólo el remitente, el intermediario de mensajes y los receptores designados. En lugar de comunicarse entre sí, las aplicaciones se comunican sólo con el intermediario de mensajes. Una aplicación envía un mensaje al intermediario de mensajes, proporcionando los nombres lógicos de los receptores. El intermediario de mensajes busca las aplicaciones registradas bajo esos nombres lógicos y luego pasa el mensaje a ellos. (Microsoft Corporation, 2010)

Este patrón está basado principalmente en la construcción de una aplicación middleware que se encarga de entender la lógica individual de cada aplicación. Aunque resuelve y simplifica muchos problemas no es una solución óptima a emplear debido a su complejidad a la hora de implementarlo.

1.5.5 Publish/Subscribe

Amplía la infraestructura de comunicación, generando tópicos o mediante inspeccionado dinámico del contenido del mensaje. Para que una aplicación consumidora pueda consumir un mensaje que le compete, dicha aplicación tiene que subscribirse a un tópico específico para escuchar únicamente los mensajes asociados a dicho tópico. (Microsoft Corporation, 2010)

Existen tres variantes

- Basado en listas.
- Publicación y Subscripción basado en una notificación masiva.
- Publicación y Subscripción basado en el contenido del mensaje.

Este patrón está basado en la subscrición y publicación de tópicos donde el intercambio de mensajes es el método de integración. La solución ofrecida no es la ideal para ser utilizada en el polo donde el intercambio de información resulta fundamental y donde no es necesaria la subscrición a un componente para poder intercambiar información con él.

1.5.6 Gateway

Gateway es un componente que abstrae el acceso a recursos externos. En caso de que se tengan diversas aplicaciones desarrolladas en distintas plataformas y necesitan comunicarse con recursos externos, en vez de que cada aplicación acceda a dichos recursos o servicios por cuenta propia, se crea una entidad de enlace la cual tiene conocimiento del lugar en que los servicios o recursos se encuentran. (Microsoft Corporation, 2010)

Este patrón sugiere un gran esfuerzo de mantenimiento y construcción donde un cambio implica mucho trabajo si se pusiera en práctica como método de integración en el polo. Por tanto, no es una solución viable a emplear.

1.6 SOA

Existen muchísimas definiciones para SOA en Internet pero no hay acuerdo común para explicar lo que este acrónimo significa. Sitios web como SearchWebServices.com han anunciado concursos para buscar las mejores definiciones pero las posibilidades de selección para la mejor y más completa definición son pocas porque SOA significa cosas diferentes para diferentes personas.

A continuación se presenta la tabla 1 donde se relacionan algunos puntos de vista planteados por el arquitecto Boris Lublinsky en sitio de IBM: (Lublinsky, Boris, 2007)

Desde el punto de vista del:	SOA es:
Ejecutivo de negocios y analista de negocios	Un conjunto de servicios que constituyen los valores de IT (capacidades) y puede ser utilizado para soluciones de construcción y exponiéndolos a los clientes y socios.
Arquitecto de empresa	Un conjunto de principios arquitectónicos y patrones para abordar las características generales de las soluciones: la modularidad, encapsulamiento, acoplamiento flexible, la separación de las responsabilidades, la reutilización, componibilidad y así sucesivamente.
Jefe de proyecto	Un enfoque de desarrollo que apoya el desarrollo masivo en paralelo.

Probador	Una manera de modularizar y simplificar, todo el sistema global de pruebas.
Desarrollador de Software	Un modelo completo de programación con estándares, herramientas y tecnologías, tales como los servicios Web.

Tabla 1: Visiones de SOA (Lublinsky, Boris, 2007)

Si bien estas visiones son definitivamente correctas el problema se encuentra en que ninguna propone una definición que internacionalmente describa a esta arquitectura de software.

Microsoft también ofrece su visión de lo que considera que es SOA:

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

A continuación se podrá ver un análisis del principal elemento de SOA, los servicios.

1.6.1 ¿Qué es un servicio en SOA?

El elemento fundamental de SOA lo constituyen los servicios, a continuación se muestran algunas definiciones planteadas internacionalmente de servicio:

Un servicio Web es un sistema de software diseñado para soportar la interoperabilidad de interacción máquina de interoperabilidad interacción persona-máquina en una red. Cuenta con una interfaz descrita en un formato procesable por máquina (específicamente WSDL). Otros sistemas interactúan con el servicio Web en la forma prescrita por su descripción usando mensajes SOAP, típicamente transmitido

usando HTTP con una serialización XML en relación con otras web relacionadas con las normas. (Booth, y otros, 2004)

Un servicio es una funcionalidad concreta que puede ser descubierta en la red y que describe tanto lo que puede hacer como el modo de interactuar con ella. Desde la perspectiva de la empresa, un servicio realiza una tarea concreta: puede corresponder a un proceso de negocio tan sencillo como introducir o extraer un dato como “Código del Cliente”. Pero también los servicios pueden acoplarse dentro de una aplicación completa que proporcione servicios de alto nivel, con un grado de complejidad muy superior –por ejemplo, “introducir datos de un pedido”-, un proceso que, desde que comienza hasta que termina, puede involucrar varias aplicaciones de negocio. (Microsoft Corporation, 2006)

La tecnología principal para la construcción de sistemas basados en SOA son los servicios Web. Los servicios Web están centrados principalmente en dos objetivos: (Building Systems using a Service Oriented, 2005)

- El cable de protocolos de nivel de ejecución que garanticen la interoperabilidad entre los sistemas heterogéneos. Los servicios web se basan en el intercambio de mensajes utilizando una tecnología de formato XML neutro (SOAP).
- Estándares para la definición de las interfaces de servicio (WSDL y XML), independientemente de la tecnología de implementación. Un estándar para la definición de las interfaces también permite la interoperabilidad entre herramientas.

1.6.2 Elementos de SOA

En la figura 12 se muestran los elementos que podrían observarse en una arquitectura orientada a servicios. (Alvez, y otros, 2006)

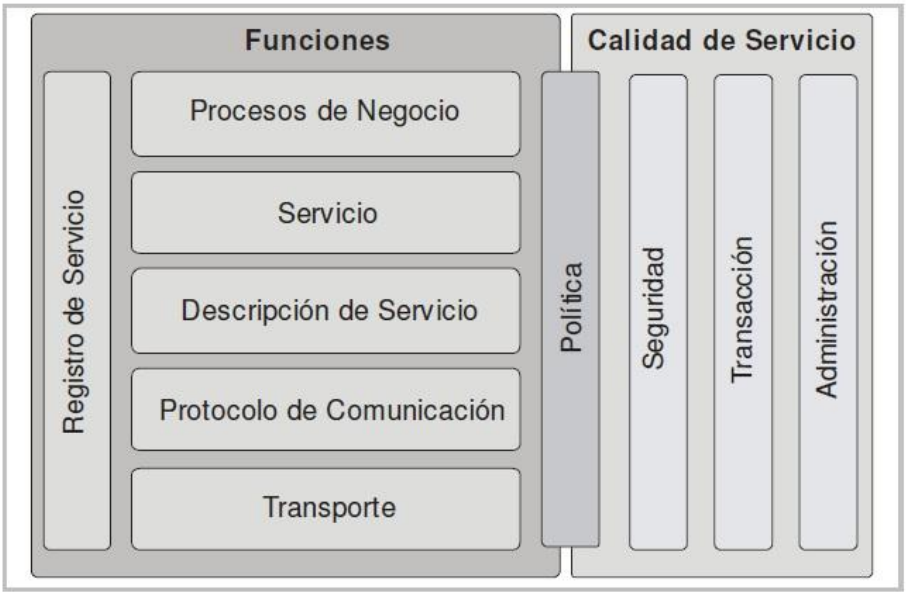

Figura 7: Elementos de SOA (Alvez, y otros, 2006)

Como se puede observar, se diferencian dos zonas una que abarca los aspectos funcionales de la arquitectura y otra que abarca aspectos de calidad de servicio. A continuación se describen los elementos brevemente: (Alvez, et al., 2006)

Funciones

- **Transporte:** es el mecanismo utilizado para llevar las demandas de servicio desde un consumidor de servicio hacia un proveedor de servicio, y las respuestas desde el proveedor hacia el consumidor.
- **Protocolo de comunicación de servicios:** es un mecanismo acordado a través del cual un proveedor de servicios y un consumidor de servicios comunican qué está siendo solicitado y qué está siendo respondido.
- **Descripción de servicio:** es un esquema acordado para describir qué es el servicio, cómo debe invocarse, y qué datos requiere el servicio para invocarse con éxito.
- **Servicios:** describe un servicio actual que está disponible para utilizar.

- **Procesos de Negocio:** es una colección de servicios, invocados en una secuencia particular con un conjunto particular de reglas, para satisfacer un requerimiento de negocio.
- **Registro de Servicios:** es un repositorio de descripciones de servicios y datos que pueden utilizar proveedores de servicios para publicar sus servicios, así como consumidores de servicios para descubrir o hallar servicios disponibles.

Calidad de Servicio

- **Política:** es un conjunto de condiciones o reglas bajo las cuales un proveedor de servicio hace el servicio disponible para consumidores.
- **Seguridad:** es un conjunto de reglas que pueden aplicarse para la identificación, autorización y control de acceso a consumidores de servicios.
- **Transacciones:** es el conjunto de atributos que podrían aplicarse a un grupo de servicios para entregar un resultado consistente.
- **Administración:** es el conjunto de atributos que podrían aplicarse para manejar los servicios proporcionados o consumidos.

Las colaboraciones en SOA siguen el paradigma find, bind and invoke, donde un consumidor de servicios realiza la localización dinámica de un servicio consultando el registro de servicios para hallar uno que cumpla con un determinado criterio. Si el servicio existe, el registro proporciona al consumidor la interfaz de contrato y la dirección del servicio proveedor. (Alvez, et al., 2006)

En la figura 13 se ilustra las entidades (roles, operaciones y artefactos) en una arquitectura orientada a servicios donde estas colaboran:

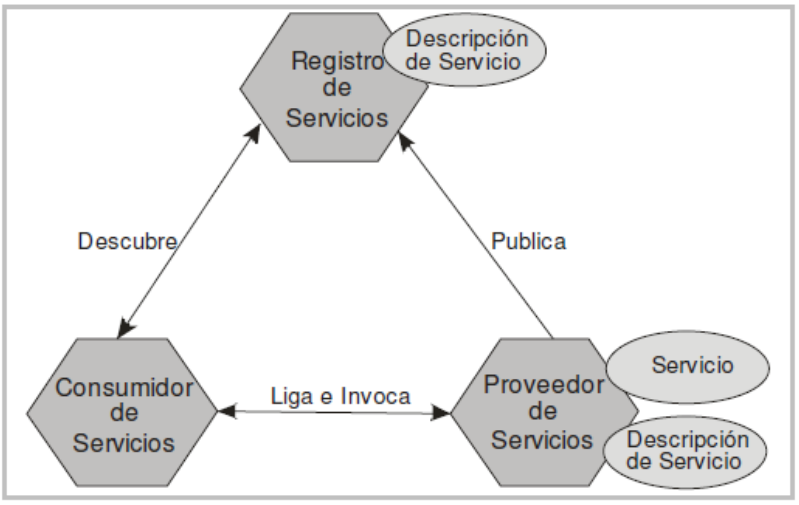

Figura 8: Colaboraciones en SOA (Alvez, y otros, 2006)

Cada entidad puede tomar el rol de consumidor, proveedor y/o registro: (Alvez, et al., 2006)

- Un consumidor de servicios es una aplicación, un módulo de software u otro servicio que requiere un servicio, y ejecuta el servicio de acuerdo a un contrato de interfaz.
- Un proveedor de servicios es una entidad direccionable a través de la red que acepta y ejecuta consultas de consumidores, y publica sus servicios y su contrato de interfaces en el registro de servicios para que el consumidor de servicios pueda descubrir y acceder al servicio.
- Un registro de servicios es el encargado de hacer posible el descubrimiento de servicios, conteniendo un repositorio de servicios disponibles y permitiendo visualizar las interfaces de los proveedores de servicios a los consumidores interesados.

Las operaciones son: (Alvez, et al., 2006)

- **Publicar.** Para poder acceder a un servicio se debe publicar su descripción para que un consumidor pueda descubrirlo e invocarlo.
- **Descubrir.** Un consumidor de servicios localiza un servicio que cumpla con un cierto criterio consultando el registro de servicios.

- **Ligar e Invocar.** Una vez obtenida la descripción de un servicio por parte de un consumidor, éste lo invoca de acuerdo a la información en la descripción del servicio.

Los artefactos en una arquitectura orientada a servicios son: (Alvez, y otros, 2006)

- **Servicio.** Un servicio que está disponible para el uso a través de una interfaz publicada y que permite ser invocado por un consumidor de servicios.
- **Descripción de servicio.** Una descripción de servicio especifica la forma en que un consumidor de servicio interactuará con el proveedor de servicio, especificando el formato de consultas y respuestas desde el servicio. Esta descripción también puede especificar el conjunto de pre-condiciones, pos-condiciones y/o niveles de calidad de servicio.

1.6.3 Propiedades de SOA (Marinelli, y otros, 2007)

Algunas de las más importantes propiedades de SOA son:

- **Orientación a la conversación:** el foco de atención no está en los nodos sino en los mensajes que se intercambian entre los mismos.
- **Vista Lógica:** El servicio es una abstracción (vista lógica) de los programas, bases de datos, procesos de negocio, etc.
- **Orientación a Mensajes:** el servicio se define formalmente en términos de los mensajes que se intercambian entre agentes proveedores y solicitantes. Esto posibilita que se incorpore un componente "decorando" estos componentes con software de gestión y conversión.
- **Abstracción del agente:** la estructura interna del agente (lenguaje de programación, BD, etc.) se abstrae en SOA, un nodo es una entidad computacional que participa en conversaciones con otros nodos. No es necesario conocer las particularidades del lenguaje de implementación. Esto evita problemas arquitectónicos derivados de la necesidad de conocer determinados sistemas a nivel estructural.

- **Metadatos:** SOA se asocia con metadatos, los mismos son descripciones acerca de la forma y tipo de los elementos que transportan los mensajes, el orden de los mensajes, el significado de los mensajes, etc.
- **Orientación a la Internet:** los servicios tienden a usarse a través de la red, aunque este no es un requisito absoluto.
- **Granularidad:** los servicios tienden a usar un pequeño número de operaciones con mensajes complejos.
- **Neutral a la Plataforma:** los mensajes se envían en un formato estándar y neutral a la plataforma, utilizando XML

1.6.4 ¿Por qué usar SOA?

Existen varias razones para que el polo adopte un enfoque SOA, y más concretamente un enfoque SOA basado en servicios web para integrar componentes de software:

- **Reutilización**: Un principio fundamental de SOA es la reutilización. Al adoptar esta arquitectura el polo podrá reutilizar las funciones de negocios que identifique dentro de las empresas para las cuales trabaja, además que pueden ser expuestas como servicios, ser reutilizadas para cubrir nuevas necesidades de negocio.
- **Interoperabilidad:** SOA resuelve los problemas presentes en el polo de interoperabilidad, con su aplicación no importará el lenguaje de construcción de cualquier componente para utilizar sus funcionalidades, debido a que estas funcionalidades serán expuestas como servicios, independientes de plataforma, lenguaje de programación y sistema operativo.
- **Escalabilidad:** En SOA los servicios están acoplados débilmente por tanto las aplicaciones que hacen uso de estos servicios en el polo podrán escalar fácilmente. Esto es debido a la poca dependencia que existe entre las aplicaciones clientes y los servicios usados por estas.
- **Flexibilidad:** Los servicios que serán implementados en el polo entre otras características poseerán un débil acoplamiento entre ellos, por tanto, los cambios hechos en los componentes no

afectarían las aplicaciones u otros servicios que los usen siempre que se mantenga la interfaz de los componentes.

- **Eficiencia de coste:** Entre las características más importantes de SOA se encuentra la eficiencia en los costes de producción, esto se debe a que los servicios son construidos una sola vez y sus funcionalidades son expuestas para que sean utilizadas. El polo no necesitaría reconstruir una nueva funcionalidad cada vez que la necesite sino reutilizar las que ya tiene consumiéndolas como servicios.

1.7 Conclusiones Parciales

Como conclusión del presente capitulo se pueden obtener las siguientes valoraciones:

- El DSBC constituye una idea revolucionaria en el negocio de construcción de aplicaciones de software.
- Trabajar siguiendo un enfoque basado en componentes significa avanzar un peldaño más en la mejora de los procesos de desarrollo de software.
- Los beneficios del DSBC con respecto a calidad, tiempo de construcción, retorno sobre la inversión entre otros indicadores son considerables.
- SOA es una idea promisoria en la que actualmente se trabaja de manera infatigable. Muchas empresas prestigiosas apuestan por esta arquitectura, resulta importante estudiarla para tomar los aspectos positivos e incorporarlos adaptándolos a las condiciones del polo.
- Si se integran el DSBC y SOA se puede obtener una infraestructura lo suficientemente sólida para soportar soluciones informáticas de alta calidad.

Capítulo II: Estrategia para la integración y publicación de componentes en el polo productivo PetroSoft

2.1 Introducción

En el presente capítulo se relacionan los patrones, técnicas, tecnologías y un grupo ordenado de pasos para lograr con éxito publicar e integrar los componentes de software en el polo productivo PetroSoft utilizando servicios.

2.2 Vista de la Arquitectura planteada por la estrategia

La presente estrategia fue concebida para aplicarse en el polo productivo PetroSoft. La misma ofrece la descripción y guía objetiva para la construcción de cada una de las partes de un sistema general de integración y publicación de las funcionalidades de los componentes como servicios web. De cada una de las partes involucradas en el sistema por la estrategia se especifica la tecnología, estándares y herramientas necesarias para que su construcción y puesta en funcionamiento se realice de forma satisfactoria.

Características del modelo planteado:

- **El diseño de la estrategia se encuentra en el nivel 4 de integración.** La visión del sistema planteado es visto desde la óptica de una empresa, donde la integración es implementada de forma estándar para que entornos exteriores interactúen con el entorno del sistema abstrayéndose de su construcción y centrándose solamente en la comunicación.
- **Se utiliza el patrón Entity Aggregation para dar solución al problema.** Este patrón presenta una solución fácil de desarrollar e implantar donde no es necesario un alto grado de complejidad. Introduce una capa entre el repositorio de componentes y el usuario final controlando toda la lógica del negocio, centralizando el flujo de información y estandarizando la comunicación.

- **La lógica del sistema es dividida en tres fundamentales: el repositorio de componentes, el repositorio de servicios y la UDDI (Universal Description, Discovery and Integration).** Es importante la utilización de cada uno de estos nodos por el papel desempeñado en el proceso de integración y publicación. Aunque en la vista arquitectónica (figura 14) puede verse un nodo adicional (Cliente), no es necesaria una explicación de las tecnologías, herramientas o plataformas para su construcción debido a que estos aspectos quedan a consideración de los usuarios del sistema y no constituye objetivo del presente trabajo.
- **Implementa una separación entre lo concerniente a implementación de componentes y lo concerniente a sistema de ensamblado.** Los componentes de software se encuentran físicamente en el repositorio de componentes lugar al cual acceden las aplicaciones servidoras que se encuentran en el repositorio de servicios e instancian estos componentes publicando sus funcionalidades como servicios. La implementación de un componente es un proceso que se lleva por separado al proceso de integración presente en el repositorio de servicios.
- **Provee una comunicación e implementación del sistema basada en estándares de comunicación.** Todo el desarrollo de la infraestructura del sistema está diseñado para desarrollar y usar estándares aprobados internacionalmente, facilitando la comunicación con el mismo y permitiendo la interoperabilidad con diferentes sistemas informáticos.
- **Define un modelo abstracto de implementación para el acceso a los componentes. Este modelo soporta la implementación de componentes en una amplia variedad de lenguajes de programación.** El sistema planteado funciona de manera general como un solo servidor que recibe las invocaciones de los servicios y devuelve las respuestas. La realidad nos muestra que no es un solo servidor sino un conjunto de servidores implementados con diferentes tecnologías que son capaces de instanciar los componentes que se encuentran en el repositorio de componentes. Cada servidor planteado tiene la capacidad de analizar las funcionalidades de los componentes dinámicamente y exponerlas como servicios. El dinamismo de estos servidores está dado por el hecho de que ellos no tienen definidas cuáles son las funcionalidades de ningún componente de antemano, sino que las definen en tiempo de ejecución cuando instancias los componentes.

La figura siguiente muestra la vista arquitectónica de la estrategia que se plantea. En ella se destacan como elementos fundamentales el repositorio de servicios, el repositorio de componentes, la UDDI y un nodo cliente que representa cualquier aplicación que invoque un servicio del repositorio de servicios.

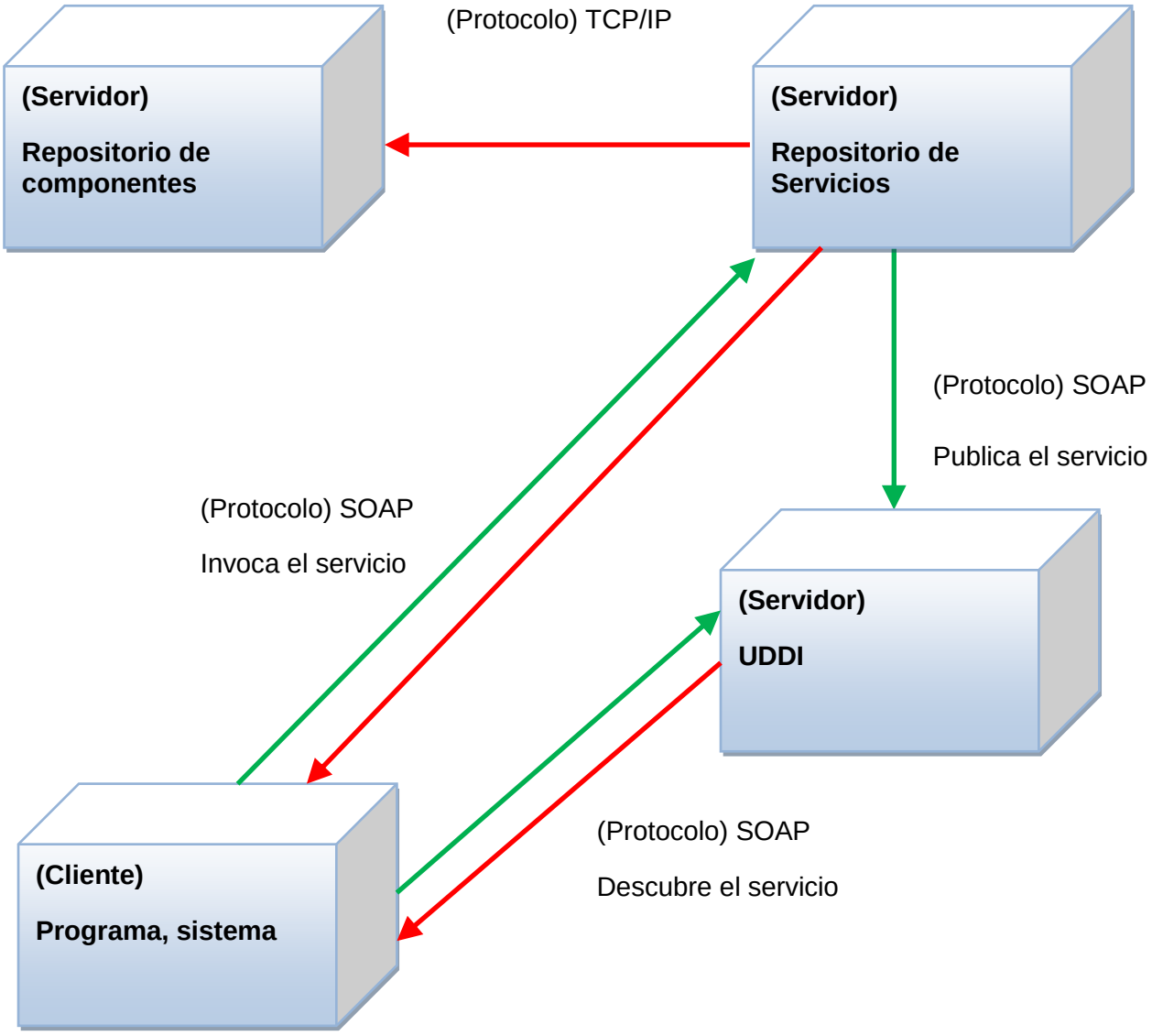

Figura 9: Vista arquitectónica de la estrategia

Para comprender cada parte de la infraestructura que se plantea, se realizará en la continuación de este capítulo una explicación de los nodos mostrados en la vista arquitectónica del sistema y se describirán

aspectos importantes relacionados con las tecnologías y herramientas escogidas para implementar la presente solución. El factor esencial para la rápida construcción y desarrollo de este sistema está dado por la comprensión de las partes que en el mismo se involucran.

2.3 Repositorio de componentes

El repositorio de componentes no es más que un sitio centralizado donde se almacena o se centralizan componentes de software, brindando además, los servicios asociados para el manejo de estos componentes de software contenidos en él.

Los repositorios son valorados como herramienta imprescindible del DSBC por la gran importancia de todas las facilidades y los servicios que brindan. Un repositorio de componentes le permitirá al equipo de desarrollo del polo almacenar, compartir y utilizar componentes desarrollados, conocer los componentes en existencia y facilitar la labor de encontrar los componentes apropiados para el sistema que se desee implementar.

El polo cuenta con una propuesta de un repositorio de componentes "*GForce*" al cual se le han realizado algunas modificaciones para adaptarlo a las necesidades productivas del polo, por tanto, no será necesario proponer una implementación de un repositorio de componentes debido a que el polo cuenta con la anteriormente comentada propuesta, pero si resulta importante describir los objetivos del repositorio de componentes pues resulta de vital importancia.

2.3.1 Objetivos del repositorio de componentes

El repositorio de componentes se ha planteado luego de analizar los problemas presentes en la producción del polo como vía estratégica para solucionar estas dificultades, el mismo tiene entre sus objetivos fundamentales los siguientes:

- No repetir esfuerzos en la producción.
- Contribuir a reducir el tiempo de entrega de los proyectos.
- Contar en todo momento, con componentes de calidad para utilizarlos en cualquier etapa del ciclo de vida del software, pues se estarían mejorando constantemente.

– Contribuir a obtener productos más acabados y mejores, teniendo en cuenta que todos los componentes que se incluyan en la biblioteca deben estar certificados por calidad.
– Permitir la socialización del conocimiento, lo cual se inscribe en el estilo de trabajo colaborativo que le interesa potenciar a la máxima dirección de la producción de software en la UCI.

2.4 Repositorio de servicios

La construcción de una aplicación que actúe como servidor de servicios web capaz de consumir componentes del repositorio de componentes independientemente del lenguaje con el cual fue construido resulta muy complejo de concebir y de desarrollar. Luego de un profundo análisis se determinó construir una aplicación servidora en cada lenguaje en los cuales el polo desarrolla aplicaciones de software y ubicar estas aplicaciones en un solo servidor.

El resultado obtenido será el mismo debido a que los servicios ofrecidos son transparentes para el usuario y la descripción para poderlos invocar se encuentra expuesta en la UDDI. Este método para la construcción del servidor de servicios web es mucho más sencillo de diseñar e implementar porque la lógica de cada aplicación servidora solo encierra el lenguaje requerido para su implementación.

Los servicios web constituyen la pieza fundamental de la presente estrategia, siendo los encargados de realizar las operaciones requeridas para resolver el problema de integración y de publicación de componentes de software.

2.4.1 Implementación de los servicios web

Los servicios web serán implementados utilizando una arquitectura formada por cinco tipos de capas de tecnologías que organizarán su construcción. A continuación se presenta la vista y una breve descripción de las capas implicadas en el desarrollo de los servicios web que serán implementados en el polo:

Figura 10: Vista de las capas implicadas en el desarrollo de los servicios del polo.

Cada capa de este modelo de construcción de servicios web separa un problema del modelo de negocio, con la implementación solamente de estas capas hace que el modelo servicio web tenga sentido. Como meta principal este modelo plantea una modularización total de los ambientes de desarrollos distribuidos.

2.4.1.1 Descubrimiento

Esta capa proveerá un mecanismo a los consumidores para buscar la descripción del proveedor acerca de los servicios web publicados. Con este propósito se plantea más adelante la construcción de una UDDI, uno de los más reconocidos mecanismos de descubrimiento.

2.4.1.2 Descripción

Cuando se realice la implementación de un servicio web cualquiera se toman decisiones en cada capa para su construcción, como la red, el transporte, los protocolos de empaquetado que lo soportan. Una descripción de ese servicio no es más que la representación de esas decisiones de manera que un consumidor pueda utilizar ese servicio. Como lenguaje de descripción de los servicios se utilizará como estándar Web Service Description Language (WSDL).

2.4.1.3 Empaquetado

Para que los datos de la aplicación se puedan mover en la red por la capa de transporte, estos deben ser empaquetados en un formato que pueda ser entendido por todas las partes interesadas. Con este fin se utilizará SOAP como protocolo para codificar esos mensajes y datos debido a que presenta ventajas de codificación sobre otros protocolos como HTTP porque está fuertemente ligado a la presentación de la información, más que a su significado.

2.4.1.4 Transporte

En la capa de transporte se incluyen las tecnologías que posibilitan la comunicación directa entre la aplicación y los servicios web. El rol principal de esta capa será el de mover los datos entre una o más localizaciones en la red. Como protocolo para este fin se ha elegido HTTP, porque provee el soporte de cortafuegos más común, es un protocolo ampliamente estandarizado y es el contenedor más utilizado para el protocolo SOAP.

2.4.1.5 Red

En la capa de red la tecnología de servicios web es exactamente la misma que la capa de red en el modelo red TCP/IP. Proporciona la comunicación de base crítica, la dirección, y capacidades de enrutamiento.

Luego de haberse explicado lo concerniente a la construcción de los servicios web se dará paso a puntualizar los datos de las librerías o frameworks utilizados en la construcción de cada servidor de servicios web.

2.4.2 Librerías para construir los servidores SOAP

Para cada lenguaje de programación se han construido a nivel internacional librerías o frameworks para el trabajo con el protocolo estandarizado SOAP. Muchas de estas implementaciones son maduras y están respaldadas en su desarrollo por prestigiosas compañías o proyectos desarrolladores de software como Microsoft y Apache, otras han quedado varadas o se ha descontinuado su desarrollo por diversas razones. De las librerías o frameworks escogidas se relacionan aspectos esenciales a continuación:

2.4.2.1 Librería gSOAP para C++

La herramienta gSOAP realiza una conexión de lenguaje entre el protocolo SOAP y el lenguaje de programación C/C++ proporcionando el desarrollo de servidores y clientes de servicios web en este lenguaje. Permite un uso transparente de SOAP basado en un compilador de interfaces que genera: Stubs

para el lado del cliente, esqueletos para el lado del servidor y código para transformar datos de tipo C/C++ a XML y viceversa.

Características: (The Florida State University, 2010)

- Distribuido bajo licencias de código libre.
- Multiplataforma (Windows Linux, Solaris).
- Interoperable con servicios y clientes programados en otras plataformas.
- Compatible con programas muti-hilos.
- Compatible con SOAP 1.1 y SOAP 1.2.
- Es auto-contenida lo que implica que contiene todo lo que necesita para funcionar.

Ventajas de gSOAP: (The Florida State University, 2010)

- **Portabilidad:** soporta la mayoría de las plataformas, incluyendo sistemas empotrados y pequeños sistemas operativos (por ejemplo, WinCE, Symbian y Palm OS). La portabilidad es probada para Windows (98, XP, Vista), Linux, Unix, Mac OS X, Solaris, HP-UX, AIX, FreeBSD, Tru64, Irix, QNX, y VxWorks.
- **Estabilidad:** es software maduro. El desarrollo y prueba se ha llevado a cabo durante varios años desde el 2001. Este software es utilizado por muchos de los proyectos y productos industriales actuales.
- **Amplia gama de usuarios:** más de 250.000 descargas desde el año 2003, más de 6.000 miembros de la lista de correo, y miles de licencias o acuerdos de apoyo con empresas, entre ellas varias compañías Fortune 100.
- **Todo en un solo paquete:** la independencia de los fabricantes de herramientas y bibliotecas de terceros garantiza el éxito y se basa en un tiempo de ejecución confiable.
- **Open source:** opciones libres de la licencia comercial.
- **Soporte C y C++:** es compatible con ANSI C y C/C++ para el desarrollo de aplicaciones.
- **Completa los enlaces de datos XML:** gSOAP es el único conjunto de herramientas SOAP/XML que soporta la conversión de datos puramente nativos de C/C + + a de datos XML. Este conjunto

de herramientas serializa automáticamente las estructuras de datos basadas en punteros, incluyendo gráficos cíclicos, y soporta STL (en parte), las enumeraciones y la herencia de clases.

- **Interoperabilidad y cumplimiento:** el conjunto de herramientas sigue las recomendaciones del estándar WS-I Basic Profile 1.0a, 1,1 y 1,2. Se alerta sobre los problemas de interoperabilidad potenciales antes de la construcción de un nuevo servicio de aplicación web, así que no se tiene que ir a través de otro ciclo de desarrollo para hacer compatible sus servicios.
- **Utiliza protocolos estándar de la industria:** SOAP 1.1/1.2, WSDL 1.1 y UDDI v2. Soporta los tipos primitivos de esquemas XML.
- **Protocolos de transporte que soporta:** HTTP/S, TCP, UDP, MIME, DIME (streaming), MTOM (streaming), HTTP1.0/1.1, IPv4, IPv6, RSS, XML-RPC, WS-Addressing, WS-Enumeration y muchos otros protocolos. HTTP POST stack soporta HTTP/1.1 POST/GET SOAP/XML de mensajería con la compresión, fragmentación, mantenimiento de la conexión, el logging, y el cifrado SSL.
- **Seguridad:** HTTPS y WS-Security: autenticación, tokens y las firmas digitales.
- **Velocidad:** la generación específica de código por el compilador es rápida. Comparaciones demuestran velocidades comparables o mejores que los demás analizadores XML (información aportada por estudios revisados por especialistas técnicos). El tiempo de ida y vuelta de los mensajes SOAP en una invocación de servicio están por debajo de 1ms.
- **Numerosos ejemplos:** el paquete de software incluye ejemplos de código, incluyendo HTTP/1.1y HTTPS.
- **Integración con servidores web:** incluye interfaces Apache_mod, IIS, WinInet, CGI y FastCGI para integrar sus servicios.

2.4.2.2 Framework .NET para C#

.NET Framework es un componente integral de Windows que admite la creación y la ejecución de la siguiente generación de aplicaciones y servicios Web XML. (Microsoft Corporation, 2007)

.NET Framework contiene dos componentes principales: Common Language Runtime y la biblioteca de clases de .NET Framework. Common Language Runtime es el fundamento de .NET Framework. (Microsoft Corporation, 2007)

La biblioteca de clases, es una completa colección orientada a objetos de tipos reutilizables que se pueden emplear para desarrollar aplicaciones que abarcan desde las tradicionales herramientas de interfaz gráfica de usuario (GUI) o de línea de comandos hasta las aplicaciones basadas en las innovaciones más recientes proporcionadas por ASP.NET, como los formularios Web Forms y los servicios Web XML. (Microsoft Corporation, 2007)

En la figura siguiente se muestra la relación de Common Language Runtime y la biblioteca de clases con las aplicaciones y el sistema en su conjunto. En la ilustración se representa igualmente cómo funciona el código administrado dentro de una arquitectura mayor.

Figura 11: NET Framework en contexto (Microsoft Corporation, 2007)

Los servicios web están generados en la parte superior de .NET Framework y el Common Language Runtime. Un servicio web puede aprovecharse de estas tecnologías. Por ejemplo, el rendimiento, administración de estados y autenticación admitidos por ASP.NET pueden aprovecharse generando servicios web con ASP.NET. (Microsoft Corporation, 2007)

La infraestructura para los servicios web está generada para cumplir con los estándares de la industria como SOAP, XML y WSDL y esto permite a los clientes de otras plataformas interoperar con servicios web. Con tal de que un cliente pueda enviar los mensajes SOAP conforme con los estándares, con un formato de acuerdo a una descripción del servicio, ese cliente puede llamar un servicio web creado con ASP.NET (sin tener en cuenta la plataforma en la que el cliente reside). (Microsoft Corporation, 2007)

Al generar un servicio web mediante ASP.NET, se admite automáticamente clientes que se comunican mediante los protocolos de SOAP, HTTP-GET y HTTP-POST. Desde que HTTP-GET y HTTP-POST permiten pasar los mensajes en pares de nombre y valor con codificación URL, la compatibilidad del tipo de datos para estos dos protocolos no está tan enriquecida como aquel compatible con SOAP. En SOAP, que pasa los datos hacia y desde el servicio web utilizando XML, se pueden definir tipos de datos complejos mediante los esquemas XSD, los cuales son compatibles con un conjunto más enriquecido de tipos de datos. Los programadores que generan un servicio web mediante ASP.NET no tienen que definir explícitamente los tipos de datos complejos que esperan mediante un esquema XSD. Más bien, simplemente pueden generar una clase administrada. ASP.NET administra las definiciones de clase de asignación en un esquema XSD y asigna las instancias de objeto a los datos XML para pasarlo de uno a otro por una red. (Microsoft Corporation, 2007)

Es importante tener en cuenta que los servicios web no son una sustitución de DCOM, sino una infraestructura de mensajería para comunicarse entre plataformas utilizando los estándares de la industria.

2.4.2.3 Librería Apache Axis2 para Java

Para usar la librería Apache AXIS se requiere, además, Tomcat (el contenedor para Servlets de Apache) y Xerces (analizador XML de Apache).

Apache Axis2 provee muchas características, mejoras y especificaciones de las implementaciones de la industria. Las principales características que ofrece son los siguientes: (Apache Software Foundation, 2009)

- **Velocidad:** Axis2 utiliza su propio modelo de objetos y StAX (Streaming API para XML) de análisis para lograr de manera significativa una mayor velocidad.

- **AXIOM:** Axis2 viene con su propio modelo ligero de objetos, AXIOM, para el procesamiento de mensajes el cual es ampliable, de alto rendimiento y es el desarrollador conveniente.
- **Despliegue en caliente:** Axis2 está equipado con la capacidad de desplegar los servicios Web y manejarlos mientras el sistema está en funcionamiento. En otras palabras, los nuevos servicios se pueden añadir al sistema sin tener que apagar el servidor. Simplemente se colocan los archivos de los servicios necesarios en el directorio web de servicios en el repositorio, y el modelo de despliegue automáticamente implantará el servicio y lo pondrá a disposición para su uso.
- **Servicios web asincrónicos:** Axis2 ahora es compatible con los servicios web asincrónicos y con la invocación asincrónica de los servicios Web usando el no bloqueo de clientes y transportes.
- **Soporte MEP:** Axis2 ahora viene de la mano con la flexibilidad necesaria para soportar los patrones de intercambio de mensajes (MEP), con el soporte básico de construcción definido en WSDL 2.0.
- **Flexibilidad:** La arquitectura Axis2 le da al desarrollador la libertad completa de insertar extensiones en el motor para el procesamiento del encabezado personalizado, gestión del sistema, y cualquier cosa que puedas imaginar.
- **Estabilidad:** Axis2 define un conjunto de interfaces publicadas que evolucionan con lentitud en comparación con el resto de Axis.
- **Orientada a componentes de implementación:** Usted puede definir fácilmente reutilizables redes de controladores para aplicar patrones comunes de procesamiento para sus aplicaciones, o para distribuir a los socios.
- **Framework de transporte:** Se tiene una abstracción limpia y simple para la integración y el uso de transportes (es decir, los remitentes y los oyentes de SOAP se comunican a través de varios protocolos como SMTP, FTP, middleware orientado a mensajes, etc.), y el núcleo del motor es totalmente independiente del transporte.
- **Soporte para WSDL:** Axis2 soporta el WSDL, versión 1,1 y 2,0, lo que permite crear fácilmente stubs para acceder a los servicios remotos, y también para exportar automáticamente las descripciones legibles por las máquinas de los servicios desplegados con Axis2.
- **Add-ons:** Algunas especificaciones para los servicios Web se han incorporado como WSS4J para la seguridad (Apache de Rampart), Sandesha para la mensajería confiable, Kandula que es una encapsulación de WS-Coordination, WS-AtomicTransaction y WS-BusinessActivity.

- **Composición y extensibilidad:** Módulos y fases para mejorar el apoyo en la composición y extensibilidad. Módulos que soportan la composición y que puede soportar también nuevas especificaciones de servicios web en una forma limpia y simple. Son sin embargo, no desplegable en caliente como los cambios en el comportamiento global del sistema.

Características de la última versión: (Apache Axis2 Versión 1.4)

- Soporte para clientes y servicios POJO y Spring.
- Soporte para cualquier patrón de intercambio de mensajes.
- Llamadas síncronas y asíncronas.
- Aprovechamiento del modelo de despliegue de servicios soportando por completo la encapsulación del servicio con el soporte de versiones.
- Aprovechamiento del modelo de implementación soportando la extensibilidad controlada con el apoyo de versiones.
- Despliegue en caliente.
- WS-Policy impulsado por las extensiones de generación de código.
- Flexible vida del ciclo del modelo de servicio.
- Soporte automático para los POX (REST) estilo de invocación de servicios.
- Apoyo para la consulta de un servicio WSDL (usando WSDL), esquema (usando XSD) y políticas (usando Policy).
- WSDL 2.0.
- Implementaciones personalizadas.
- Serialización binaria (FastInfoset).
- Soporte para JSON.
- Soporte para el proveedor EJB.

2.4.2.4 SOAP.py para Python

SOAP.py es una biblioteca donde se encuentran todas las funciones requeridas para el trabajo con SOAP en Python. Tiene como meta principal un uso fácil de la misma y una total compatibilidad con la dinámica de las interacciones entre los clientes y los servidores.

Incluye: (Ullman, y otros, 2005)

- Analizador general basado en SOAP sax.xml.
- Constructor general SOAP.
- Proxy SOAP para el código de clientes RPC.
- Framework del servidor SOAP para el código del servidor RPC.

Características: (Ullman, y otros, 2005)

- Maneja todos los tipos de SOAP 1.0.
- Maneja las fallas.
- Permite la especificación de espacio de nombres.
- Permite la especificación SOAPAction.
- Permite matrices con tipo homogéneo.
- Soporta múltiples esquemas.
- Soporte para Header.
- Soporte para el atributo XML.
- Soporte multi-referencia (analizador / constructor).
- Entiende SOAP-ENC: atributo raíz.
- Buena interoperabilidad, pasa todas las pruebas del cliente para la Frontier, SOAP::Lite, SOAPRMI.
- Soporta codificaciones.
- Soporta clientes SSL (con Python compila con soporte OpenSSL).
- Soporta servidores SSL (con Python compila con soporte OpenSSL y M2Cryptoinstalado).
- Codifica etiquetas XML SOAP 1.2.
- Soporta estado automático del servidor SOAP (Apache v2.x) (blunck2).
- Soporte para cliente WSDL.
- Soporte para servidor WSDL.

2.4.2.5 Librería NUSOAP para PHP

La librería NUSOAP ofrece un conjunto de herramientas para implementar servicios web en el lenguaje de programación PHP. En su composición pueden verse varias clases que permiten hacer un uso más fácil de la librería para la construcción de los servicios web. Esta herramienta provee soporte para el desarrollo de clientes y servidores de servicios web. NUSOAP tiene soporte para SOAP 1.1, WSDL 1.1 y HTTP 1.1/1.

Aunque, existen varias librerías para el trabajo con SOAP y la implementación de servicios web con el lenguaje PHP, NUSOAP se ofrece como uno de los proyectos que más avanzados se encuentra. PHP 5 a partir de su versión 5 empieza a dar soporte para el trabajo con SOAP, pero aún es un proyecto nuevo y se encuentra en fase experimental. NUSOAP ofrece tres características fundamentales que la hacen ideal para el desarrollo de servidor de servicios para el lenguaje PHP:

- Está en una fase madura de desarrollo.
- No necesita módulos adicionales.
- Es muy fácil su instalación y uso.

2.5 UDDI

UDDI es el acrónimo de Universal Description, Discovery and Integration, catálogo de negocios de Internet. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el ámbito de los servicios Web. Proporciona un conjunto de normas basadas en estándares de la industria del software para la descripción y descubrimiento de servicios.

No existen requisitos de propietarios respecto a la manera en que un desarrollador implemente su nodo UDDI, cualquier compañía o empresa puede implementar su propio directorio de servicios UDDI. El nodo sólo se debe ajustar a la especificación UDDI. Por lo que nodos que utilicen tecnologías diferentes a la hora de su implementación, su comportamiento será el mismo ante los clientes, ya que se ajustan a un mismo conjunto de llamadas API XML basadas en SOAP. Las herramientas de los clientes pueden inter-operar con los diferentes nodos sin problemas.

2.5.1 Diseño Propuesto

Se propone la implementación de una UDDI con una topología de tipo “Nodo Privado UDDI”, este proveerá la funcionalidad de UDDI a través de la Intranet a los usuarios del polo, este registro no será accesible desde Internet ni por ninguna otra organización, por lo que sus datos no se replicarán con ningún otro servidor UDDI. Este tipo de topología trae consigo que se puedan definir las propias políticas de seguridad y acceso. De esta manera, el polo contará con una excelente herramienta para la publicación, búsqueda y categorización de sus servicios web que serán reutilizados por parte de los desarrolladores de aplicaciones.

El desarrollo de la UDDI será guiado por el diseño planteado en especificaciones de UDDI 3.0.2, definidas por OASIS. En este diseño se explica claramente el desempeño de las estructuras y funcionalidades de UDDI, que han sido sujetas a mejoras en cada una de sus versiones. La UDDI es diseñada como un registro no como un depósito. Esta diferencia resulta esencial. Un registro redirige al usuario a los recursos, mientras que un depósito solo almacena información.

2.5.2 Especificaciones de la UDDI

Teniendo en consideración la especificación de la UDDI plantada por OASIS deben describirse dos APIs, la API de pregunta y la API de publicación, estas API difieren en los documentos en XML, las estructuras de datos y los puntos de acceso.

La API de pregunta sirve para la localización de información acerca de los negocios, los servicios ofrecidos por estos, las especificaciones de estos servicios e información sobre qué hacer cuando se produce algún fallo. Con esta API cualquier operación de lectura del registro UDDI se efectúa con mensajes de esta API. Esta API no requiere acceso autenticado, razón por la cual los accesos se realizan mediante el protocolo de transporte HTTP.

La API de publicación se usa para crear, almacenar o actualizar información ubicada en un registro UDDI. En esta API, todas las funcionalidades expuestas necesitan que el acceso sea autenticado. El registro UDDI debe tener identidades para autenticarse cuando se necesite entrar al sistema y las credenciales de seguridad deben pasarse como parámetros en cada invocación a la UDDI en el documento XML. Por tanto el protocolo de transporte utilizado para conectarse a esta API es HTTPS, pero utilizando una dirección URL diferente de la usada en acceso al punto de las preguntas.

2.5.2 Estructura de los datos en la UDDI

La información de UDDI se encuentra dividida en tres secciones fundamentales (páginas blancas, amarillas y verdes), en las cuales se almacenan los datos de sus principales estructuras, a continuación se muestran en la figura 17:

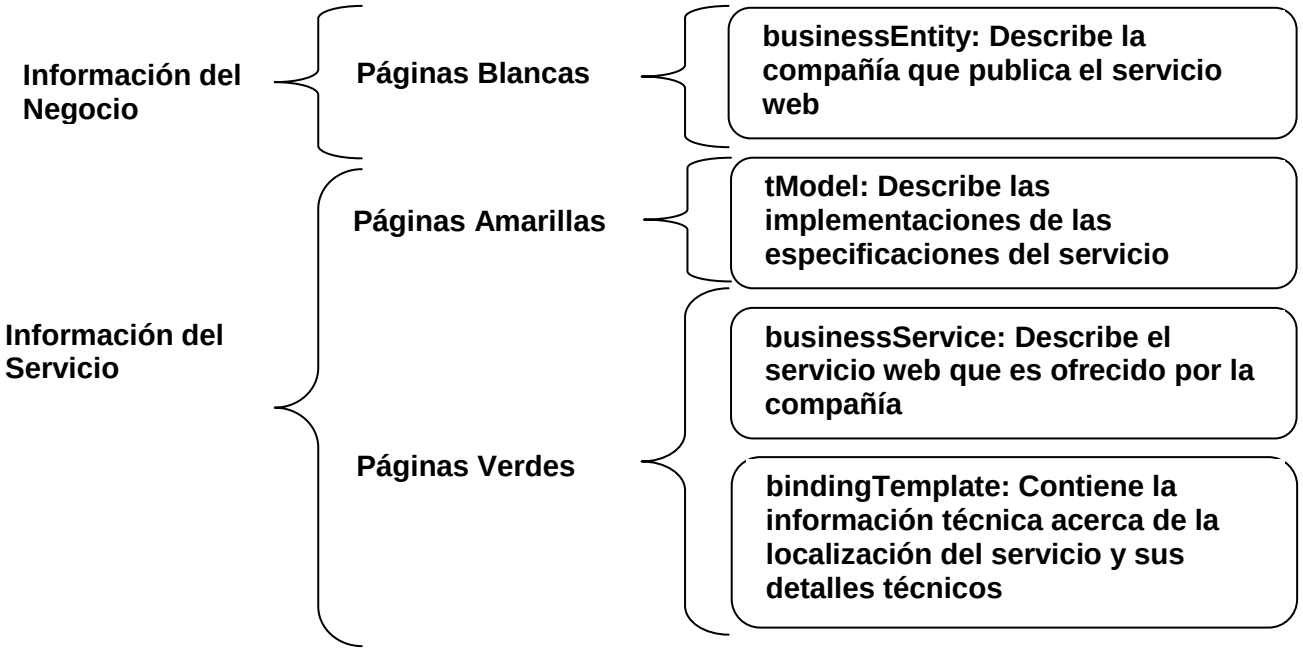

Figura 12: Tipos de datos de la UDDI

La UDDI debe tener definidos los cuatro tipos básicos de datos para la información del negocio y del servicio. Todos los datos de la UDDI deben pertenecer a uno de estos cuatro tipos de datos mostrados anteriormente. UDDI define cada tipo en una estructura de datos basadas en XML y cada uno contiene campos obligatorios y opcionales. Los cuatro tipos de datos son:

businessEntity: La información referente a cualquier unidad empresarial se encontrará almacenada en este tipo de estructura. La información contenida en la misma permitirá buscar y localizar los datos de las empresas, estos datos están contenidos en las páginas amarillas y blancas. Este tipo de estructura puede contener una o varias estructuras **businnessService.**

businessService: Los servicios o procesos de negocios que suministra la estructura **businessEntity** son representados por esta estructura. Es utilizada para agrupar una serie de servicios relacionados que presentan analogías con los procesos empresariales o con las categorías de servicios. También permite la categorización de los servicios por el tipo de industria, producto, servicio o frontera geográfica. Este tipo de estructura puede contener una o varias estructuras binfingTemplate.

bindingTemplate: Las características técnicas de la implementación de los servicios ofrecidos son representados por esta estructura que incluye además información de los puntos de acceso a los servicios y la descripción de estos servicios. Contiene una lista de referencias a la información referente a las especificaciones que forman la huella digital técnica de los servicios y establecen qué comportamiento o interfaz de programación implementa el servicio. Estas referencias son una colección de claves que se pueden utilizar para tener acceso a información de metadatos acerca de una especificación, en la forma de un elemento denominado **tModel**.

tModel: El rol fundamental de esta estructura es la de representar una especificación técnica. Contiene metadatos que definen una especificación de servicio, incluido el nombre, la información acerca del editor y los punteros URL a la especificación técnica real que define al servicio, lo que permite que numerosos servicios se adapten al mismo **tModel** registrado. En el caso de un servicio web, **tModel** apuntará al archivo WSDL correspondiente al servicio web. Juntos, **businessService**, **bindingTemplate** y **tModel**, aportan información a los datos de las páginas verdes acerca de los servicios proporcionados por las empresas.

2.5.3 Importancia de la construcción de la UDDI:

Las razones fundamentales que hacen de la UDDI un nodo con gran importancia en el presente trabajo son:

- Brinda una solución escalable para organizar, detectar, volver a usar y administrar servicios web.
- Presenta esquemas de categorización para describir proveedores y sus servicios que se pueden personalizar para satisfacer las necesidades de cualquier organización.
- Integración con diversas herramientas de desarrollo.
- Administración fácil de usar con el complemento Servicios UDDI.

2.5.4 Lenguaje de construcción propuesto

Para la construcción de la UDDI se analizaron tres lenguajes de programación, la siguiente tabla arroja los resultados obtenidos:

Criterios	PHP	Java	ASP
Plataforma	Multiplataforma	Multiplataforma	Plataforma única
Conocimientos en el polo	Familiarizados	Poco conocimiento	Poco conocimiento
Licencia	Es un lenguaje libre	Es un lenguaje libre	Uso bajo licencia propietario
Velocidad de ejecución	Alta	Baja	Alta
Recursos	Bajo consumo	Alto consumo	Bajo consumo
Integración con servidores web	Fácil integración con la mayoría de los servidores web	Fácil integración con la mayoría de los servidores web	Solamente soportado por Internet Information Server (IIS)

Tabla 2: Comparación entre tres lenguajes de programación.

Es fácilmente observable que PHP se presenta como la mejor solución de lenguaje para la construcción de la UDDI. También es necesaria la creación de un portal web para hacer legible la información de la UDDI, de esta manera, los usuarios podrán realizar búsquedas e informarse de una manera fácil. Es importante señalar algunas ventajas más de PHP que lo hace la mejor solución a emplear.

Ventajas de PHP: (Ródenas, 2007)

- Es un lenguaje de alto nivel que no requiere definición de tipos de variables.
- Es libre y de código abierto, por lo que se presenta como una alternativa de fácil acceso para todos.

- Permite leer y manipular datos desde diversas fuentes, incluyendo datos que pueden ingresar los usuarios desde formularios HTML.
- Es un lenguaje multiplataforma ya que tiene la capacidad de ser ejecutado en la mayoría de sistemas operativos.
- Puede interactuar con los servidores web más populares, ya existe en versión CGI, módulo para Apache e ISAPI.
- Dispone de una gran capacidad de conexión con la mayoría de los manejadores de base de datos que se utilizan en la actualidad, aunque destaca su conectividad con MySQL.
- Tiene una gran capacidad de expandir su potencial utilizando la enorme cantidad de módulos o extensiones que dispone.
- Posee una amplia documentación en su página oficial en la que se incluye una biblioteca de funciones sumamente amplia.
- Posibilita crear aplicaciones con interfaz gráfica para el usuario, utilizando la extensión GTK. También puede ser usado desde la línea de comandos, de la misma manera que Perl o Python pueden hacerlo. Permite las técnicas de programación orientada a objetos.

2.6 Comunicación y estándares utilizados

En la tabla 3 que se muestra a continuación se relacionan los protocolos y estándares utilizados en la concepción de la estrategia:

Estándares	
Descripción de los servicios	WSDL
Registro y búsqueda de los servicios	UDDI
Uso de los servicios	SOAP
Formato estándar de los datos	XML
Comunicación	HTTP

Tabla 3: Tablas de estándares utilizados

A continuación se describen brevemente las principales características de los estándares utilizados en la presente propuesta, los mismos constituyen la base para el desarrollo de una infraestructura web basada en estándares internacionalmente utilizados y estandarizados.

2.6.1 WSDL

Es un formato basado XML que se utiliza para describir los servicios web. Fue creado por Microsoft e IBM, es el lenguaje utilizado en la UDDI para la descripción de la interfaz pública de los servicios web. Por tanto, describe la forma de comunicación como los protocolos y los formatos de los mensajes necesarios para interactuar con los servicios listados en el catálogo servicios de la UDDI. Las operaciones y mensajes que soporta se describen en abstracto al protocolo concreto de la red y al formato de los mensajes. Los documentos WSDL utilizan los siguientes elementos en la definición de servicios de red:

- ***types*:** proveen definiciones de los tipos de datos utilizados para describir los mensajes intercambiados.
- ***message*:** que representa una definición abstracta de los datos que están siendo transmitidos. Un mensaje se divide en una serie de partes lógicas, cada una de las cuales se asocia con alguna definición de algún sistema de tipos.
- ***portType*:** es un conjunto de operaciones abstractas. Cada operación hace referencia a un mensaje de entrada y uno de salida.
- ***binding*:** especifica un protocolo concreto y las especificaciones del formato de los datos de las operaciones y los mensajes definidos por un por *Type* en concreto.
- ***port*:** especifica una dirección para un *binding*, para así definir un único nodo de comunicación.
- ***service*:** se utiliza para unir un conjunto de puertos relacionados.

En el documento WSDL se definen cuatro tipos de operaciones:

- **Request-response (*petición-respuesta*):** Comunicación en la que el cliente realiza una petición y el servidor envía la correspondiente respuesta.
- **One-way (*un-sentido*):** Comunicación del estilo documento en la que el cliente envía un mensaje pero no recibe una respuesta del servidor indicando el resultado del mensaje procesado.

- **Solicit-response (*solicitud-respuesta*):** El servidor envía una petición y el cliente le envía de vuelta una respuesta.
- **Notification (*Notificación*):** El servidor envía una comunicación del estilo documento al cliente.

2.6.2 SOAP

La figura mostrada a continuación muestra los principales elementos de los mensajes SOAP:

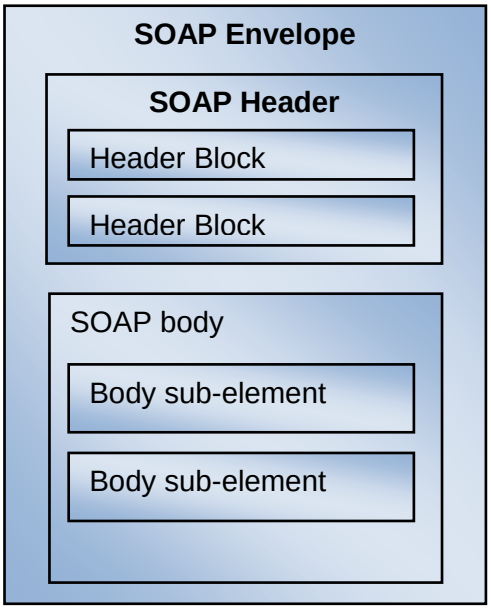

Figura 13: Estructura de los mensajes SOAP

SOAP es un protocolo estandarizado creado principalmente por Microsoft e IBM, actualmente se encuentra bajo el auspicio de W3C donde se define cómo dos procesos diferentes pueden comunicarse mediante el empaquetado de los mensajes intercambiados. Las especificaciones de SOAP lo definen como una simple envoltura basada en XML para la información que se transmite y un grupo de reglas para la representación y traslado específico de datos en aplicaciones y plataformas.

Un mensaje SOAP se compone de una envoltura que contiene un encabezado opcional y un cuerpo obligatorio. La cabecera contiene bloques de información pertinente a la forma en que el mensaje debe ser procesado. Esto incluye la configuración de la expedición y de la entrega, la autenticación o aseveraciones de autorización, y los contextos de transacciones. El cuerpo contiene el mensaje que

deberá ser entregado y procesado. Cualquier cosa que se pueda expresar en sintaxis XML puede ir en el cuerpo de un mensaje.

Es importante señalar que una de las ventajas de utilizar SOAP sobre HTTP es que HTTP es un protocolo basado en un convenio de solicitud-respuesta. El mensaje de solicitud SOAP se envía al servidor HTTP con la petición HTTP y el servidor devuelve el mensaje de respuesta SOAP en la respuesta HTTP.

Ventajas ofrecidas por SOAP:

- No está asociado con ningún lenguaje.
- Puede transportarse utilizando cualquier protocolo capaz de transmitir texto, ya que no es más que un documento XML.
- Aprovecha los estándares existentes en la industria.
- Permite la comunicación entre entornos diferentes.

2.6.3 XML

Es un Lenguaje de Etiquetado Extensible muy simple, pero estricto que juega un rol fundamental en el intercambio de gran variedad de datos. Fue estandarizado por la W3C y ofrece una sintaxis sencilla para etiquetar documentos de forma legible. Presenta un formato flexible que se utiliza en gráficos vectoriales (SVG), formatos de documentos multimedia (Open Document Formator Applications, Open Document), formatos para intercambio electrónico de datos (B2B), serialización de objetos, codificación de parámetros y métodos en los RPC (RPC-XML). Los códigos XML se pueden crear y modificar mediante editores de texto. Algunas aplicaciones pueden manejar sólo algunos tipos de documentos XML, mientras que otras pueden interpretar cualquier tipo de XML.

Características de XML

Existen distintas bibliotecas para su lectura, creación, manipulación, traducción en otros formatos, en distintas plataformas. XML añade “inteligencia” a los datos que se envían. Por sí solos, no tienen significado.

Características de XML:

- XML añade meta-información, etiquetas autodescriptivas para cada parte del texto, de forma que el documento incluye el contenido y su descripción.
- Un conjunto de elementos XML relacionados forman un vocabulario.
- La instancia de un vocabulario se llama documento XML, que es el bloque fundamental de XML.
- Las distintas organizaciones que quieran intercambiar información con el formato XML pueden acordar el conjunto estándar de vocabularios, o proporcionar tecnologías para realizar la traducción de unos vocabularios a otros.
- No sólo los datos son textos, sino también las etiquetas. Esto permite que sea portable.
- Los vocabularios pueden declararse formalmente con un lenguaje de esquema XML.

2.6.4 HTTP

Es un conjunto de estándares que les permite a los usuarios de la web intercambiar información. Es el método que se utiliza para transferir documentos desde el sistema donde se almacenan las páginas hasta los usuarios individuales.

Características de HTTP:

- Protocolo de transferencia de hipertexto.
- Utiliza protocolo de transporte TCP, y el puerto estándar del servicio es el 80.
- Es un protocolo orientado a transacciones y sin estados.
- Para poder direccionar un recurso en la web se utiliza el Localizador Universal de Recurso (URL, Universal Resource Locator).
- Posee mensajes en texto ASCII.

Tiene dos métodos básicos:

- GET: Solicitud para obtener la información identificada en la URL, obteniéndola en el cuerpo de la entidad.
- POST: Solicitud para admitir la entidad adjunta como subordinada a la URL indicada.

2.7 Comunicación

A continuación se explicarán los pasos básicos en la comunicación y trabajo con la infraestructura planteada:

El primer paso es el consumo de un componente de software desde el repositorio de servicios. Las aplicaciones servidoras que residen en el repositorio de servicios se conectan al repositorio de componentes y consumen los componentes de software para conocer sus funcionalidades y exponerlas como servicios web. Es importante señalar que los componentes deben encontrarse físicamente en el repositorio de componentes a donde se accede utilizando el protocolo TCP/IP que permite la entrega de datos sin errores y en el orden que fueron trasmitidos. El segundo aspecto a tener en cuenta es que las aplicaciones servidoras solamente instancian estos componentes cuando se invoca el respectivo servicio, mientras el servicio no es invocado las aplicaciones servidoras no consumen los componentes permitiendo un eficiente uso de la memoria y evitando la congestión de las redes. Por último debe existir un archivo de configuración al cual las aplicaciones servidoras accedan para conocer los detalles necesarios para comunicarse con el repositorio de componentes, estos detalles son útiles pues permiten conocer el IP del repositorio de componentes así como datos necesarios para establecer la comunicación, de esta manera cuando se cambien los datos para acceder al repositorio de componentes solamente será necesario editar este archivo de configuración.

El segundo paso es publicar los servicios en la UDDI de tal manera que los usuarios puedan conocer la existencia de los mismos y puedan invocarlos. Una vez que los servicios son expuestos es importante que las aplicaciones servidoras se conecten con la UDDI y expongan las características de esos servicios, publicando el archivo WSDL donde se encuentran las descripciones de los mismos. Para lograr lo anteriormente expuesto las aplicaciones servidoras harán uso de las funcionalidades ofrecidas por cada librería para el trabajo con los documentos WSDL. En este proceso de comunicación entre las aplicaciones servidoras y la UDDI se utilizará el protocolo estándar SOAP.

El tercer paso es el descubrimiento por parte de los usuarios de los servicios que han sido expuestos en la red. Cuando un usuario necesita utilizar un servicio se conecta con la UDDI para realizar una búsqueda entre los servicios ofrecidos y de esta manera definir cuál de ellos le resulta más útil dependiendo de su necesidad. En este proceso de comunicación se utilizará el protocolo estándar SOAP.

El cuarto paso es la invocación de un servicio que se encuentra expuesto en la red. Luego que el usuario ha descubierto el servicio lo invoca para hacer uso del mismo utilizando en este proceso de comunicación el protocolo estandarizado SOAP. Este proceso de invocación de un servicio provoca que la aplicación servidora que ofrece este servicio obtenga los datos del mensaje SOAP, luego se conecte al repositorio de componentes consume el componente que tiene la funcionalidad requerida y le pase los datos obtenidos, el componente procesa estos datos y devuelve una respuesta, la aplicación servidora toma esta respuesta la convierte en un mensaje SOAP y envía este mensaje como respuesta al cliente que invocó el servicio.

Básicamente estos son los cuatro pasos más importantes involucrados en los procesos de comunicación en la infraestructura planteada. Los usuarios son los encargados de construir los mensajes para comunicarse con el repositorio de servicios y la UDDI.

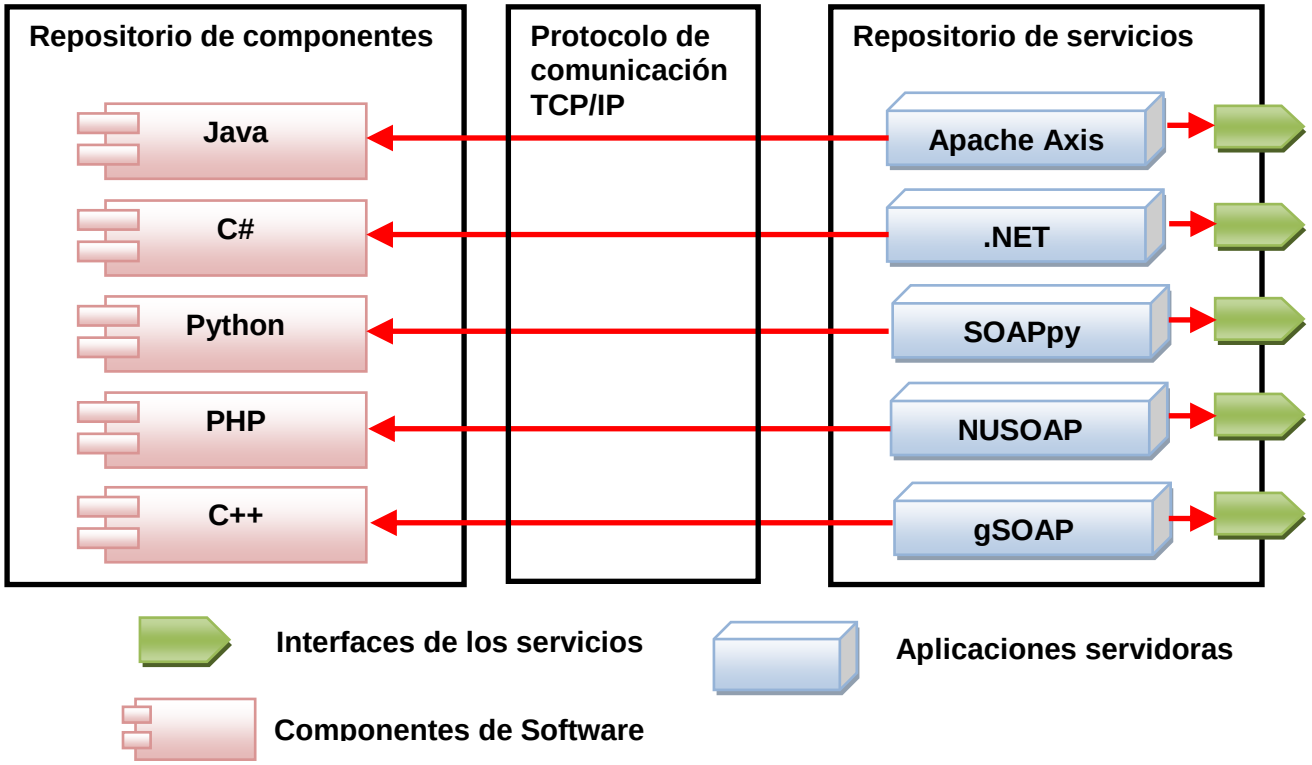

Figura 14: Comunicación entre repositorio de componentes y el repositorio de servicios.

La figura anterior muestra cómo quedará conformada la estructura en el repositorio de servicios y el repositorio de componentes y la comunicación entre ellos.

2.8 Ventajas y desventajas

A continuación se relacionan las ventajas y desventajas de la estrategia planteada.

2.8.1 Ventajas del modelo que se plantea

- Reutiliza las funcionalidades planteadas por el repositorio de componentes, debido a que no es necesario la construcción o una modificación del repositorio de componentes del polo para lograr el sistema que se quiere implantar sino que el repositorio de componentes continuará brindando las funcionalidades para lo cual fue concebido.
- Reutiliza las funcionalidades de los componentes que se encuentran en el repositorio de componentes del polo. Dicho de otra manera quiere decir que cuando una aplicación requiera utilizar las funcionalidades ofrecidas por algún componente que se encuentre en el repositorio de componentes del polo solo tendrá que consumir el servicio web de este componente. De esta manera, el implementador no tiene que preocuparse de la lógica ni de la implementación de este componente.
- Se pueden implementar componentes que den soluciones a problemas existentes en el lenguaje que más fácil o conveniente sea debido a que estas funcionalidades pueden ser expuestas como servicios web.
- Provee al usuario de una abstracción total de los componentes usados.
- Los cambios o actualizaciones de los componentes no suponen problema para los usuarios porque este proceso es transparente para ellos.
- El diseño de servicios basados en estándares facilita la creación de un repositorio de servicios reutilizables que se pueden combinar en servicios de mayor nivel y aplicaciones compuestas en respuesta a nuevas necesidades del polo. Con ello se reduce el coste del desarrollo de soluciones y de los ciclos de prueba, se eliminan redundancias y se consigue su puesta en valor en menos tiempo. Y el uso de un entorno y un modelo de desarrollo unificados simplifica y estandariza la creación de aplicaciones, desde su diseño y prueba hasta su puesta en marcha y mantenimiento.
- Desarrollo de aplicaciones más rápido y económico. Aplicaciones más seguras y manejables. Las soluciones orientadas a servicios proporcionan una infraestructura común (y una documentación común también) para desarrollar servicios seguros, predecibles y gestionables.

2.8.2 Desventajas del modelo que se plantea

- Introduce una capa adicional de arquitectura que pueda afectar negativamente el rendimiento final de las soluciones.
- La gestión de un gran número de instancias de proceso en una aplicación central puede presentar un cuello de botella en tiempo de ejecución.

2.9 Conclusiones parciales

Como conclusión del capítulo se pueden obtener las siguientes:

- La infraestructura planteada resuelve los problemas de integración del polo siguiendo una orientación a servicios.
- Desde el punto de vista del desarrollo de aplicaciones de software, la orientación a servicios supone un marco conceptual mediante el cual se puede simplificar la creación y mantenimiento de sistemas y aplicaciones.
- SOA es una fórmula para reutilizar e integrar los recursos necesarios manteniendo el modelo de negocio y adaptándose a las necesidades y dinámicas de cambio que les afecten.
- Es necesario un estudio continuo de las dinámicas del cambio en la producción de software dentro del polo para poder ir adaptando estos nuevos cambios a la infraestructura.

Capítulo III: Evaluación y Aplicación de la Propuesta

3.1 Introducción

Luego de haber realizado un estudio sobre las técnicas y patrones que existen a nivel internacional para realizar los procesos de integración de software y de haber definido una estrategia para la integración de componentes de software en el polo PetroSoft, se validará la propuesta realizada tomando como base el empleo de las técnicas del método de Delphi. Para utilizar la propuesta en el polo es necesario tener validada la propuesta de estrategia.

3.2 Validación de la estrategia. Método de Delphi

El método de Delphi es una técnica de investigación que tiene como fin la confección de una opinión general fidedigna a partir de las opiniones de un grupo de expertos. Su nombre es inspirado en el antiguo oráculo de Delphos y tiene su origen a inicios de los años 50 en el seno del Centro de Investigación estadounidense RAND Corporation. Fue creado por Olaf Helmer y Theodore J. Gordon como instrumento para realizar predicciones con fines militares, pero a partir de la década de los sesenta ha sido empleado en marcos de trabajos académicos y empresariales como técnica de previsión y consenso en situaciones de incertidumbre donde no es posible utilizar otras técnicas apoyadas en información objetiva.

3.2.1 Proceso de selección de los expertos

Este método no cuenta con un valor exacto que determine el valor óptimo para el número de expertos que se debe seleccionar. Los investigadores de Rand Corporation expresan que es preciso contar con un mínimo de 7 expertos o con un máximo de 30. En la presente investigación se resolvió contar con 7 expertos, teniendo en cuenta que el contenido tratado es de un alto nivel de complejidad.

Para seleccionar el grupo de expertos que darán sus criterios acerca de la presente estrategia se utilizaron personas con considerables conocimientos en temas como las metodologías de desarrollo de software, procesos de integración de software y arquitectura de software. En este grupo de expertos se encuentran líderes de proyectos, arquitectos principales de proyectos, profesores de Ingeniería de Software con años de trabajo en la profesión y programadores expertos en temas de integración. Como

resultado de una buena selección se puede obtener resultados con alto índice de eficiencia y opiniones con un elevado grado de convergencia y de aquiescencia.

3.2.2 Elaboración del objetivo a valorar

El objetivo fundamental a evaluar por los expertos es: La estrategia para publicar e integrar los componentes de software que han sido desarrollados en el polo productivo PetroSoft, referente a los elementos planteados para la integración y su aplicación en el desarrollo de software en los proyectos.

3.2.3 Aplicación del método de Delphi (Ver anexo 1 para las fases generales del método)

Paso 1: Conformar criterios de evaluación y organizarlos por grupos

El rango de evaluación establecido está entre 1 y 10, donde 1 representa el valor mínimo y 10 representa el máximo valor que puede tomar cada criterio. Se ha defino cuatro grupos generales de criterios a evaluar, dentro de estos se encuentra los sub-criterios que constituyen las categorías en evaluación, a continuación se presentan:

Criterios de mérito científico:

- C1. Calidad de la investigación.
- C2. Novedad científica.
- C3. Aporte científico.

Criterios de implantación:

- C4. Satisfacción de las necesidades en la producción.
- C5. Garantía de principios básicos de integración.

Criterios de generalización

- C6. Facilidades de comprensión.
- C7. Facilidades de uso.

- C8. Adaptabilidad a diferentes entornos de producción de software.

Criterios de impacto

- C9. Contribución al Proceso de Desarrollo de Software.
- C10. Contribución al proceso de desarrollo en proyectos productivos.

Paso 2: Asignar a cada grupo de criterios un peso relativo

Grupo	Peso relativo
No.1	30
No.2	20
No.3	30
No.4	20

Tabla 4: Representación del peso relativo de cada grupo de criterios

Paso 3: Seleccionar un comité de expertos conformado por un mínimo de 7 expertos. Esta elección estuvo basada en la especialidad, grado científico y currículum de cada uno.

Paso 4: Se entregará a cada experto la propuesta para que sea estudiada, y evaluada a través de una encuesta (Ver anexo 2).

Paso 5: Calcular por cada criterio el peso promedio, partiendo de los pesos dados por los expertos

El peso promedio por cada criterio se muestra en la siguiente tabla:

G	C/E	E1	E2	E3	E4	E5	E5	E7	EP
30	C1	10	10	10	9	10	9	10	9.71428
	C2	8	8	9	8	9	10	9	8.71428
	C3	10	8	9	9	9	10	9	9.14285
20	C4	8	8	9	9	7	10	9	8.57142
	C5	8	8	7	8	9	8	9	8.14285

30	**C6**	9	10	10	10	10	10	10	9.85714
	C7	10	9	10	10	10	9	10	9.71428
	C8	9	8	9	10	9	10	10	9.28571
20	**C9**	8	7	8	8	8	9	9	8.14285
	C10	10	10	10	8	9	9	8	9.14285
Total		90	86	91	89	90	94	93	90.42851

Tabla 4: Resultado del trabajo de los expertos

Paso 6: Determinar la consistencia en el trabajo de los expertos

La figura que se muestra a continuación muestra los criterios que fueron objeto de evaluación con los valores expresados por los expertos representados en porciento:

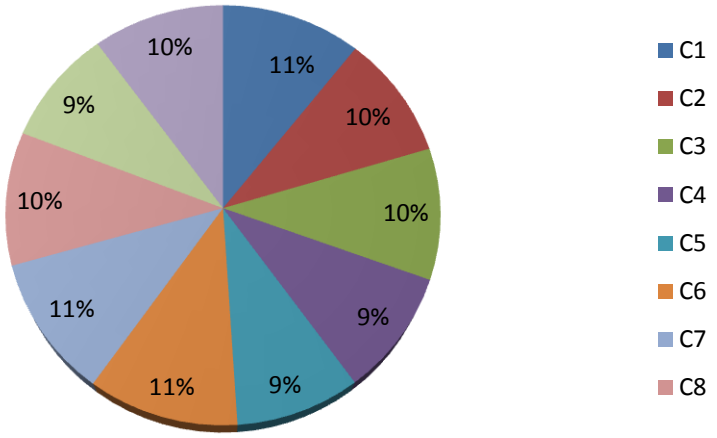

Figura 15: Representación del valor de los criterios en porciento.

Para confirmar la consistencia en el trabajo realizado por parte de los expertos seleccionados se emplea el coeficiente de concordancia de Kendall y el estadígrafo Chi cuadrado (X^2).

El procedimiento requerido para este cálculo implica:

Sea C el número de criterios que van a evaluarse y (E) el número de expertos que realizan la evaluación.

Para cada criterio se determina:

- ΣE: Sumatoria del peso dado por cada experto.
- Ep: Puntuación promedio del peso dado por cada experto.
- MΣE: media de los ΣE.
- ΔC: Diferencia entre ΣE y MΣE.

Se determina la desviación de la media, que posteriormente se eleva al cuadrado para obtener la dispersión (S) por la expresión:

$S = \Sigma (\Sigma E - \Sigma \Sigma E / C)^2$

Conociendo la dispersión se puede calcular el coeficiente de concordancia de Kendall (W):

$W = S / E^2 (C^3 – C)/12$

El coeficiente de concordancia de Kendall permite calcular el Chi cuadrado real:

$X^2 = E (C-1) W$

Los valores obtenidos se muestran en la siguiente tabla.

C/E	E1	E2	E3	E4	E5	E6	E7	∑E	EP	ΔC^2
C1	10	10	10	9	10	9	10	68	9.71428	22.09
C2	8	8	9	8	9	10	9	61	8.71428	5.29
C3	10	8	9	9	9	10	9	64	9.14285	0.49
C4	8	8	9	9	7	10	9	60	8.57142	10.89
C5	8	8	7	8	9	8	9	57	8.14285	39.69
C6	9	10	10	10	10	10	10	69	9.85714	32.49
C7	10	9	10	10	10	9	10	68	9.71428	22.09
C8	9	8	9	10	9	10	10	65	9.28571	2.89
C9	8	7	8	8	8	9	9	57	8.14285	39.69
C10	10	10	10	8	9	9	8	64	9.14285	0.49
M∑E	63.3									
S	176.1									
W	0.04356									

X^2	2.74428
$X^2(\alpha, c-1)$	21.6660

Tabla 5: Tabla para el cálculo de concordancia de Kendall

Luego de obtenido el valor del coeficiente de Kendal, se compara el X^2 real, con el valor del dato estadístico, siendo α=0.01, y C = 10 y debe cumplirse que X2 < X2 (α, c-1) para que el trabajo realizado por los expertos sea valorado de consistente. X2 (α, c-1) = X2 (0.01, 9) = 21.6660. Por tanto 2.74428 < 21.6660, quedando demostrada la consistencia en el trabajo realizado por los expertos.

Paso 7: Calcular P x c donde P es el peso relativo de cada criterio y c es la calificación promedio dada por los expertos.

Después de comprobar la consistencia del trabajo de expertos se puede definir el peso relativo de cada criterio (P).

$$P_i = E_i / \sum_{1}^{10} E$$

Conociendo el peso de cada criterio y la calificación dada por los evaluadores en una escala de 1-5 se puede construir la Tabla 5 calificación de cada criterio, para obtener el valor de P × c., donde (c), es el criterio promedio concebido por los expertos.

Criterios	Calificación					P	P x c
	1	2	3	4	5		
C1					x	0.1074	0.537
C2					x	0.0963	0.4815
C3				x		0.1011	0.4044
C4				x		0.0947	0.3788
C5					x	0.0900	0.45
C6					x	0.1090	0.545
C7				x		0.1074	0.4296
C8				x		0.1026	0.4104

C9					x	0.0900	0.45
C10					x	0.1011	0.5055

Tabla 6: Tabla de calificación de cada criterio

Paso 8: Se calcula el Índice de Aceptación (IA) de la propuesta.

IA = Σ (P × c) / 5

IA = 0.91844

Paso 9: Determinar la probabilidad de éxito de la propuesta a partir de los rangos predefinidos del índice de aceptación:

Rangos predefinidos de índice de aceptación

IA > 0,7 Existe alta probabilidad de éxito.

0,7 > IA > 0,5 Existe probabilidad media de éxito.

0,5 > IA > 0,3 Probabilidad de éxito baja.

0,3 > IA Fracaso seguro.

Por lo que se puede concluir que la propuesta tiene una alta probabilidad de éxito.

3.3 Conclusiones parciales

La evaluación de la estrategia para la publicación e integración de componentes de software a través de servicios por el método de expertos Delphi, permite validar y garantizar que el desarrollo e implantación de la misma en los proyectos productivos del polo PetroSoft se realice con éxito. Como conclusiones parciales se obtuvieron las siguientes:

- El método Delphi respondió a las necesidades de validación de la propuesta como el mejor método a emplear.
- Existe concordancia entre los expertos de que la aplicación de la propuesta realizada será un éxito e implicará números beneficios.

Conclusiones

La presente investigación ha explorado las técnicas y herramientas que se encuentran en el ámbito de integración y ha explotado las características de SOA para crear una infraestructura capaz de publicar e integrar componentes de software satisfaciendo necesidades de producción en el polo, como conclusiones se obtuvieron las siguientes:

- Permite un nuevo enfoque en el diseño y desarrollo de las aplicaciones.
- Contribuye significativamente en el aumento de la productividad, calidad, reutilización y retorno sobre la inversión; disminuyendo además el tiempo de desarrollo de las aplicaciones, el esfuerzo necesario en la construcción de soluciones informáticas y la complejidad de las soluciones.
- Introduce una forma eficiente de actualizar, corregir y agregar las funcionalidades de los componentes sin que esto implique un cambio en las aplicaciones que lo utilicen.

Recomendaciones

En el estudio realizado sobre las directrices actuales con respecto a las tendencias existentes en el desarrollo de software, podemos observar la creciente utilización de la arquitectura SOA para el desarrollo de soluciones informáticas donde se utiliza además el enfoque DSBC. Para el perfeccionamiento y continua evolución de la infraestructura anteriormente planteada se recomienda lo siguiente:

- Aplicar la presente estrategia en el polo productivo PetroSoft.
- Realizar un estudio donde se valore la posible aplicación de la estrategia presentada en otros polos productivos de la Universidad.
- Estudio y análisis del Lenguaje de Ejecución de Procesos de Negocio con Servicios Web *(*WS-BPEL*)* para introducirlo en la infraestructura planteada.
- Estudio del modelo de programación SCA para su puesta en práctica dentro de la estrategia.
- Analizar el cambio del repositorio de componentes por otro que ofrezca un mejor aprovechamiento.

Trabajos citados

Alden Szyperski, Clemens. 2010. Windows Live. [En línea] 31 de Agosto de 2010. [Citado el: 16 de febrero de 2010.] http://netbuzos.spaces.live.com/blog/cns!994BD929B80714FB!155.entry.

Alvez, Pablo, Foti, Patricia y Scalone, Marco. 2006. *Estado Del Arte. Proyecto Batuta - Generador de Aplicaciones Orquestadoras.* Urugay : s.n., 2006.

Apache Software Foundation. 2009. The Apache Software Foundation. [En línea] 2009. [Citado el: 2 de Abril de 2010.] http://ws.apache.org/axis2/.

Booth, David, y otros. 2004. W3C. [En línea] Febrero de 2004. [Citado el: 23 de Febrero de 2010.] http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

Building Systems using a Service Oriented. **Beisiegel, Michael, y otros. 2005.** 2005.

Casal Terreros, Julio. 2010. MSDN. *Sitio Web de Microsoft Corporation.* [En línea] Microsoft, 2010. [Citado el: 25 de Enero de 2010.] http://msdn.microsoft.com/es-es/library/bb972268.aspx.

IBM-WebSphere01. 2010. Windows Live. [En línea] 31 de Agosto de 2010. [Citado el: 14 de Febrero de 2010.] http://netbuzos.spaces.live.com/blog/cns!994BD929B80714FB!155.entry.

Lublinsky, Boris. 2007. IBM. [En línea] 9 de Enero de 2007. [Citado el: 25 de Febrero de 2010.] Defining SOA as an architectural style.

Marinelli, Marcelo y Kuna, Horacio. 2007. Sistemas de Información Cooperativos de la Universidad de Málaga. [En línea] Agosto de 2007. [Citado el: 11 de Enero de 2010.] http://www.sicuma.uma.es/sicuma/independientes/argentina08/Marinelli-Kuna/index.html#introduccion.

Microsoft Corporation. 2006. La Arquitectura Orientada a Servicios (SOA) de Microsoft. *Microsoft.* [En línea] 2006. http://www.microsoft.com/soa.

—. 2006. La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real. *MSDN.* [En línea] Diciembre de 2006. [Citado el: 19 de Enero de 2010.] http://www.microsoft.com/soa.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978573.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978592.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978585.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978579.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978603.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978598.aspx.

—. 2007. MSDN. [En línea] 2007. [Citado el: 23 de 5 de 2010.] http://msdn.microsoft.com/es-es/library/zw4w595w.aspx.

—. 2007. MSDN. [En línea] 2007. [Citado el: 23 de 5 de 2007.] http://msdn.microsoft.com/es-es/library/ba0z6a33.aspx.

—. 2006. MSDN. [En línea] 26 de Junio de 2006. [Citado el: 25 de Enero de 2010.] http://msdn.microsoft.com/es-es/library/ms978348.aspx.

Molina García, Juan Carlos y Ariza Rojas, Maribel. 2004. *INTRODUCCIÓN Y PRINCIPIOS BÁSICOS DEL DESARROLLO DE SOFTWARE BASADO EN COMPONENTES.* 2004.

Ródenas, Jose Ramón Belando. 2007. REPOSITORIO DIGITAL. UNIVERSIDAD POLITECNICA DE CARTAGENA. *Diseño e implementación de una herramienta para la gestión telemática de guías docentes.* [En línea] Enero de 2007. [Citado el: 1 de Abril de 2010.] http://repositorio.bib.upct.es/dspace/bitstream/10317/253/1/pfc2117.pdf.

Software Engineering Institute Carnegie Mellon. 2010. Windows Live. [En línea] 31 de Agosto de 2010. [Citado el: 16 de Febrero de 2010.] http://netbuzos.spaces.live.com/blog/cns!994BD929B80714FB!155.entry.

The Florida State University. 2010. FSU Computer Cience. [En línea] 2010. [Citado el: 25 de Marzo de 2010.] http://www.cs.fsu.edu/~engelen/soap.html.

Ullman, Cayce y Matthews, Brian. 2005. SourceForge.net. [En línea] 27 de 7 de 2005. [Citado el: 28 de 5 de 2010.] http://pywebsvcs.sourceforge.net/soappy.txt.

Bibliografía

Alden Szyperski, Clemens. 2010. Windows Live. [En línea] 31 de Agosto de 2010. [Citado el: 16 de febrero de 2010.] http://netbuzos.spaces.live.com/blog/cns!994BD929B80714FB!155.entry.

Alvez, Pablo, Foti, Patricia y Scalone, Marco. 2006. *Estado Del Arte. Proyecto Batuta - Generador de Aplicaciones Orquestadoras.* Urugay : s.n., 2006.

Apache Software Foundation. 2009. The Apache Software Foundation. [En línea] 2009. [Citado el: 2 de Abril de 2010.] http://ws.apache.org/axis2/.

Booth, David, y otros. 2004. W3C. [En línea] Febrero de 2004. [Citado el: 23 de Febrero de 2010.] http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/.

Building Systems using a Service Oriented. **Beisiegel, Michael, y otros. 2005.** 2005.

Casal Terreros, Julio. 2010. MSDN. *Sitio Web de Microsoft Corporation.* [En línea] Microsoft, 2010. [Citado el: 25 de Enero de 2010.] http://msdn.microsoft.com/es-es/library/bb972268.aspx.

IBM-WebSphere01. 2010. Windows Live. [En línea] 31 de Agosto de 2010. [Citado el: 14 de Febrero de 2010.] http://netbuzos.spaces.live.com/blog/cns!994BD929B80714FB!155.entry.

Kanerva, Pekka. 2007. *State of the art of SOAP libraries in Python and Ruby.* s.l. : Helsinki Institute for Information Technology, 2007. 1458-9478.

Lublinsky, Boris. 2007. IBM. [En línea] 9 de Enero de 2007. [Citado el: 25 de Febrero de 2010.] Defining SOA as an architectural style.

Marinelli, Marcelo y Kuna, Horacio. 2007. Sistemas de Información Cooperativos de la Universidad de Málaga. [En línea] Agosto de 2007. [Citado el: 11 de Enero de 2010.] http://www.sicuma.uma.es/sicuma/independientes/argentina08/Marinelli-Kuna/index.html#introduccion.

Microsoft Corporation. 2006. La Arquitectura Orientada a Servicios (SOA) de Microsoft. *Microsoft.* [En línea] 2006. http://www.microsoft.com/soa.

—. 2006. La Arquitectura Orientada a Servicios (SOA) de Microsoft aplicada al mundo real. *MSDN.* [En línea] Diciembre de 2006. [Citado el: 19 de Enero de 2010.] http://www.microsoft.com/soa.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978573.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978592.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978585.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978579.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978603.aspx.

—. 2010. MSDN. [En línea] 2010. [Citado el: 10 de Marzo de 2010.] http://msdn.microsoft.com/en-us/library/ms978598.aspx.

—. 2007. MSDN. [En línea] 2007. [Citado el: 23 de 5 de 2010.] http://msdn.microsoft.com/es-es/library/zw4w595w.aspx.

—. 2007. MSDN. [En línea] 2007. [Citado el: 23 de 5 de 2007.] http://msdn.microsoft.com/es-es/library/ba0z6a33.aspx.

—. 2006. MSDN. [En línea] 26 de Junio de 2006. [Citado el: 25 de Enero de 2010.] http://msdn.microsoft.com/es-es/library/ms978348.aspx.

Molina García, Juan Carlos y Ariza Rojas, Maribel. 2004. *INTRODUCCIÓN Y PRINCIPIOS BÁSICOS DEL DESARROLLO DE SOFTWARE BASADO EN COMPONENTES.* 2004.

Ródenas, Jose Ramón Belando. 2007. REPOSITORIO DIGITAL. UNIVERSIDAD POLITECNICA DE CARTAGENA. *Diseño e implementación de una herramienta para la gestión telemática de guías docentes.*

[En línea] Enero de 2007. [Citado el: 1 de Abril de 2010.] http://repositorio.bib.upct.es/dspace/bitstream/10317/253/1/pfc2117.pdf.

Software Engineering Institute Carnegie Mellon. 2010. Windows Live. [En línea] 31 de Agosto de 2010. [Citado el: 16 de Febrero de 2010.] http://netbuzos.spaces.live.com/blog/cns!994BD929B80714FB!155.entry.

The Florida State University. 2010. FSU Computer Cience. [En línea] 2010. [Citado el: 25 de Marzo de 2010.] http://www.cs.fsu.edu/~engelen/soap.html.

Ullman, Cayce y Matthews, Brian. 2005. SourceForge.net. [En línea] 27 de 7 de 2005. [Citado el: 28 de 5 de 2010.] http://pywebsvcs.sourceforge.net/soappy.txt.