RISI **1**(1), 38-48 (2004)
Rev. investig. sist. inform.
Facultad de Ingeniería de Sistemas e Informática
Universidad Nacional Mayor de San Marcos
ISSN: 1815-0268 (impreso)

SERVIDOR DE PROCESOS ADMINISTRATIVOS PARA AMBIENTES HETEROGÉNEOS USANDO CÓDIGO LIBRE (SPA)

Ulises Román, Jorge Guerra*

RESUMEN

Las Tecnologías existentes para una implementación de sistemas heterogéneos que permitirá la conectividad total entre la arquitectura propietaria (Microsoft) y las tecnologías de código libre (Java, Apache, etc.) será *WebService*(WS) el cual desempeñara un papel importante en la integración de los procesos administrativos de las unidades de negocio con las unidades corporativas de una institución pública.

El presente trabajo introduce diferentes metodologías para desarrollar un servidor de procesos administrativos para ser consultadas en diferentes plataformas, considerando dos entornos más importantes en la informática distribuida como son: Microsoft .NET y Java Web Services Development Kit (JWSDK); que apoyan en servicios Web. Estaríamos garantizando la conectividad adecuada en lo que las empresas corporativas necesitan y quieren sin necesidad de comprometerse con un producto determinado.

Palabras Claves: Webservices, procesos administrativos, código libre, Java, plataformas, sistemas distribuidos, integración, middleware, RMI, XML.

FILE SERVER FOR ADMINISTRATIVE PROCESSES IN HETEROGENOUS ENVIRONMENT USING OPEN SOURCE

ABSTRACT

The existing Technologies for an implementation of heterogenous systems that will allow the total connectivity between the propietary architecture (Microsoft) and the technologies of free code (Java, Apache, etc) will be WebService(WS) which played an important role in the integration of the administrative processes of the units of business with the corporative units of an institution publishes. The present work introduces different methodologies to develop a servant of administrative processes for consulted in different platforms, considering two important surroundings but in the distributed computing as they are: Microsoft.NET and Java Web Services Development Kit (JWSDK); that they support in services web we would be guaranteeing the suitable connectivity in which the corporative companies need and with no need want to commit themselves with acertain product.

Key words: Administrative webservices, processes, free code, distributed Java, platforms, systems, integration, middleware, RMI, XML.

1. INTRODUCCIÓN

Muchas Organizaciones se enfrentan hoy al reto de hacer negocios en un entorno cambiante y competitivo –para ello es necesario contar con las herramientas tecnológicas de información y de comunicación que les permita usar, acceder e integrar procesos administrativos en múltiples plataformas– permitiendo optimizar sus procesos administrativos y sus recursos de información *basado en una estrategia de herramientas abiertas*. En este escenario proponemos desarrollar un SPA utilizando la *tecnología de Web Services(WS)* [1] por su construcción, funcionalidad e integración de los

* Docentes de la Facultad de Ingeniería de Sistemas e Informática, Universidad Nacional Mayor de San Marcos, Lima-Perú. E-mails: {nromanc, jguerrag}@unmsm.edu.pe

procesos administrativos de las unidades de negocio con las unidades corporativas de una institución pública.

Existen diversas herramientas para la gestión de servidores y aplicaciones distribuidas en las organizaciones como: Java Application Servers (JAS) [4] – ProcessServer, Servidor AIX [5], Lam MPI y J2EE(Java) [4], etc. Cualquier aplicación puede conectarse a una aplicación J2EE utilizando la tecnología de *Web Services* (SOAP, UDDI, WSDL, XML) [2].

Para invocar y ejecutar procedimientos remotos en computadoras se ha considerado el uso de RMI(Remote Method Invocation) por su interoperabilidad, y CORBA (Common Object Request Broker Arquitecture) para invocar procedimientos en *«x» lenguaje* a partir de otro *»x» lenguaje*, así como para ejecutar método /funciones en los diversos objetos del sistemas [3].

Para el diseño y desarrollo del proyecto SPA se ha considerado el estudio de las diferentes arquitecturas que monitoree y configure la ejecución de tareas a nivel local y que permita el acceso a los usuarios remotos que provengan de un ambiente heterogéneo, como método de solución se ha utilizado *Web Services* (WS) para plataformas: Java(código libre) y .NET (propietario). Para el proceso de desarrollo de aplicaciones OOAD-UML se describen los resultados preliminares y finalmente las conclusiones del proyecto SPA.

II. FUNDAMENTACIÓN TEÓRICA

Para el desarrollo de SPA en ambientes heterogéneos usando código libre hemos considerado como fundamentos básicos a: 1. Sistemas distribuidos, 2. Tecnología Web Services, 3. CORBA-RMI, 4. Middleware, 5. J2EE y .NET, y 6. Software Libre–Linux; que a continuación describimos:

1. Sistemas distribuidos

Conocidos como «Colección de máquinas/procesos que colaborar para cumplir un objetivo» que pueden servir para *definir las arquitecturas, aplicaciones y manejo de las B.Ds que cubra el protocolo de comunicación* que puede variar dependiendo del tipo de aplicación que se quiera construir. Sin embargo, los mínimos requeridos para un ambiente de transacciones distribuidas son: *la pila de protocolo usado para la comunicación, administración de la conexión, seguridad, soporte de transacciones, marshalling y unmarshalling de datos, administración de versiones, manejo de errores, auditoria de las transacciones,* entre otros [1, 3].

Las aplicaciones llamadas COMPONENTES son ofrecidas por el contenedor para actividades como: comunicaciones, transacciones, ejecución. Como plataforma más representativa tiene a J2EE, con JSP/Servlets como tecnologías de Web Application y DNA, con ASP como tecnología Web Application en la figura N.º 1 se muestra una aplicación distribuida para procesador de transacciones donde se tiene el modelo cliente–servidor, para el caso de nuestro proyecto el cliente (alta dirección) y el servidor procesos administrativos.

2. Tecnología WebServices

Un WebService es un conjunto de aplicaciones que proporcionan datos y servicios a otras aplicaciones, sin importar las plataformas en las que están soportadas ni el lenguaje en el cual están implementadas. En forma general podemos *decir que los Web Services* [6] *«son una arquitectura de computación distribuida en evolución que usan sus propias* interfaces *programa-programa*, *protocolos* y *servicios de registro de tal manera que posibilitan que aplicacio-*

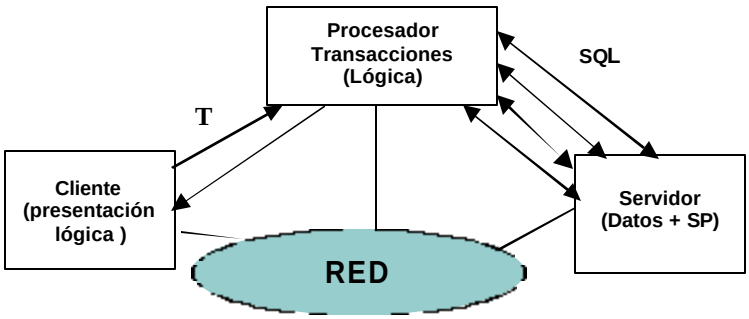


Figura N.º 1. Ejemplo de una aplicación distribuida para procesador de transacciones.

nes de diferentes plataformas tecnológicas puedan utilizar servicios de otras aplicaciones».

Los *Web Services* tienen dos propiedades adicionales: **1)** Deben tener una interfaz pública definida en una gramática común en XML, la interfase describe todos los **MÉTODOS** disponibles a los clientes y especifica la **firma** para cada método. La definición de esta interfaz se hace con el lenguaje **WSDL (Web Service Description Language), 2)** Si usted crea un servicio web, debe tener una forma de publicarlo, debe existir una forma de localizar el servicio y localizar su interfase pública**,** esto se hace con **UDDI (Universal Description, Discovery, and Integration)** en la figura N.º 2 se definen los actores, objetos y las operaciones que realiza los WS.

Los WS abordados desde cuatro macro categorías que se definen partiendo de la arquitectura o WSA.

3. RMI - CORBA

a) RMI

RMI (Remote Method Invocation) y algunas alternativas como CORBA y COM son mecanismos para *invocar o ejecutar* procedimientos remotos en *computadoras con servidores distribuidos*. Es la implementación de la idea de procedimientos remotos, y esto se debe a que la gran mayoría de los sistemas empresariales hoy en día requieren de esta funcionalidad, esto se debe tanto a distancias geográficas como a requerimientos de computo.

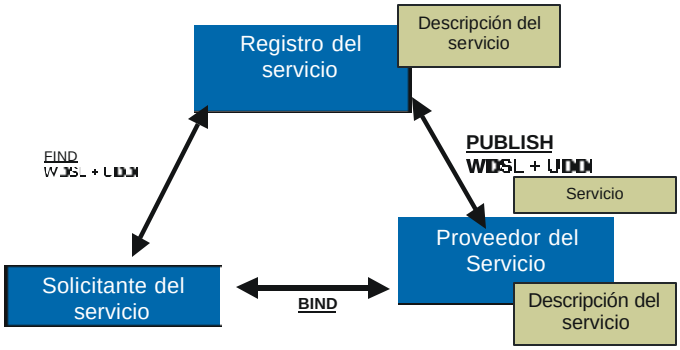


Figura N.º 2: Web services: actores, objetos y operaciones

Tabla N.º 1. Definición de Web Services

Término	Definición
Web Services Architecture (WSA)	Una aproximación estandarizada a conectividad e interoperabilidad dinámica de componentes que se ejecutan en tiempo real y bajo estándares de conectividad abierta incluyendo: Internet Protocol (IP), Simple Object Access Protocol (SOAP) y Web Services Description Language (WSDL). Otro de los estándares involucrados es el Extensible Markup Language (XML).
Software para Web Services	Incluye las herramientas de desarrollo de software, infraestructura y componentes de aplicación que conforman la WSA.
Software de desarrollo y despliegue de Web Services	Herramientas de desarrollo de software, ambiente de desarrollo e infraestructura de desarrollo.
Infraestructura de Software para Web Services	Infraestructura para ambientes de ejecución y funciones para soporte administrativo y seguridad, empaquetamiento y transmisión de mensajes y otras funciones que conforman la WSA.
Componentes de Aplicación de los Web Services	Los componentes de software de aplicación pueden usarse solos o combinados con otros componentes o aplicaciones, que son entregados sobre la red y expuestos mediante una interfaz.
Hardware de los Web Services	Está compuesto por la infraestructura de componentes de la empresa que conforman la WSA.

RMI es el mecanismo ofrecido *en Java* que permite a un procedimiento (método, clase, aplicación *o como guste llamarlo*) poder ser invocado remotamente. Una de las ventajas al diseñar un procedimiento con RMI es *interoperabilidad*, ya que RMI forma parte de todo JDK, por ende, cualquier plataforma que tenga acceso a un JDK también tendrá acceso a estos procedimientos.

b) CORBA (Common Object Request Broker Arquitecture)

CORBA al igual que varias tecnologías aceptadas hoy en día es *sólo una especificación* que fue creada en 1989 por OMG (Object Management Group). Como el nombre de la organización lo implica, CORBA establece estándares para la comunicación de *objetos* a través de procedimientos/métodos remotos.

IDL (*Interface Definition Language*)

IDL es un lenguaje utilizado para crear *cualquier* desarrollo en CORBA, su nombre es un indicador de su funcionamiento: *definición de interfases*, esto es, a través de IDL se definen las diversas estructuras que serán utilizadas en un ambiente CORBA.

```
module un ejemplo {

   interface Saludos {
  string decir Hola();
    };

};
```

El fragmento anterior es una declaración *muy sencilla* de IDL, la cual define una interfase llamada *Saludos* con un método/función llamado decir *Hola*; desde luego en IDL también pueden ser definidos cualquier estructura esperada en un lenguaje de programación: arreglos, funciones con parámetros, secuencias, etc.

En la figura N.º 5 se describe la relación que tiene los ORBs del lado del cliente como del lado del servidor.

El ORB (*Object Request Broker*) es la parte medular de un sistema CORBA, ya que a través de éste se comunican los diversos «Stubs» y «Skeletons» generados a través de IDL, es el ORB quien ofrece la conectividad en un sistema CORBA.Y al igual que todo producto depende de «especificaciones» que existen en diversos **ORB's.**

4. MIDDLEWARE

Se implementaría un puente de intercomunicación entre las distintas aplicaciones de unidades de negocio mediante la tecnología RMI o CORBA. De esta manera se llegaría a una interoperabilidad entre aplicaciones, pero con la necesidad de que tendrían que instalarse en cada una de los usuarios finales *un componente del Middleware utilizado*, vital para la comunicación. Además de ello también se tendría que realizar un mutuo acuerdo entre las unidades involucradas para establecer por mutuo consenso cual de los Middleware utilizar. Si una unidad nueva desea formar parte de la integración, esta tendría que implementar un puente de comunicación con cada una de las unidades que forman parte de la integración, esto obviamente es muy complejo y resulta muy costoso.

5. J2EE y Microsoft.NET

Tanto J2EE **(Java Enterprise Edition)** y .NET suponen la evolución de las tecnologías existentes hasta el momento para la construcción de aplica-

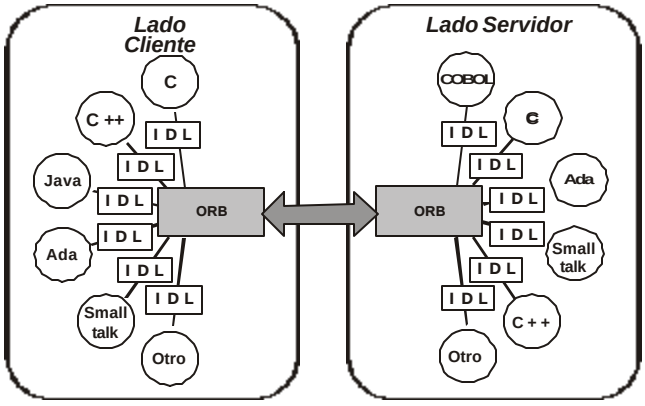


Figura N.º 5. Los ORBs de CORBA

ciones distribuidas. Las anteriores versiones de estas tecnologías no incluían la posibilidad de construir servicios web, pero ahora ambas han hecho evolucionar sus plataformas para proporcionar esta posibilidad. Podemos decir que ambas concepciones, J2EE y .NET, comparten la idea de que existe una gran cantidad de programación intrínseca a los servicios web necesaria para hacerla posible, como puede ser la interoperabilidad entre las aplicaciones distribuidas en la comunicación, el control de transacciones, la interpretación de los mensajes XML, etc. **El .NET** define un Common Language Runtime (CLR) y un IL (Intermediate Language) al que todos los lenguajes conformes a .NET compilan. Idea similar a la máquina virtual de Java y a los *bytecodes* generados por el compilador de Java, respectivamente. Lenguajes, Visual Basic .NET, Visual C++ .NET, Visual C# .NET, Visual J# .NET, etc. proporcionan una funcionalidad similar a J2EE, en particular, incluyen COM+ apuesta por servicios web como solución para interoperabilidad [4].

La arquitectura J2EE está basada en el lenguaje de programación Java. Este lenguaje de programación, bastante extendido, nació con la idea de que cualquier programa fuera escrito una sola vez y pudiera ser desplegado en cualquier plataforma sin tener que cambiar el código ni recompilar. Una vez construido el programa en lenguaje fuente Java, este es transformado a un lenguaje intermedio, *bytecode*, a medio camino de código máquina y código fuente. Este *bytecode* es ejecutado por un intérprete en tiempo de ejecución denominado JRE (*Java Runtime Environment*) [4] [8].

6. SOFTWARE LIBRE-LINUX

«Software Libre» se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software. El software libre en administraciones públicas abre una vía imparable para el desarrollo tecnológico a través de su implantación en la administración pública. La plataforma código abierto ha considerado el Sistema Operativo Linux, para la base de Datos: **MySQL**; para diseño de paginas: **PHP** y como servidor: **APACHE**. Para las consultas es necesario definir el protocolo de comunicación y accesos a cada una de las unidades de negocio y para podemos usar Lam **MPI** (*Message Passing Interface*). LAM MPI el único software de ingeniería transparente a usuarios y a sistemas administrativos se usa a nivel de *cluster* en procesamiento paralelo, usando LINUX y **KDE** como interfase gráfica, constituyéndose así como *memoria distribuida* [7].

1II. MÉTODO Y/O METODOLOGÍA

Se adaptado la metodológica de **MSF** (*Microsoft Solutions Framework*) en la parte del Modelo de Arquitectura Empresarial(MAE) debido a dos características fundamentales: primero por el concepto de integración de las unidades corporativas con respecto unidades locales (procesos, SI, internet) y el segundo por las arquitecturas(tecnológica, negocios, aplicación e información) que se visualiza en la figura N.º 7 y que se ajusta a ambientes heterogéneos y distribuidos.

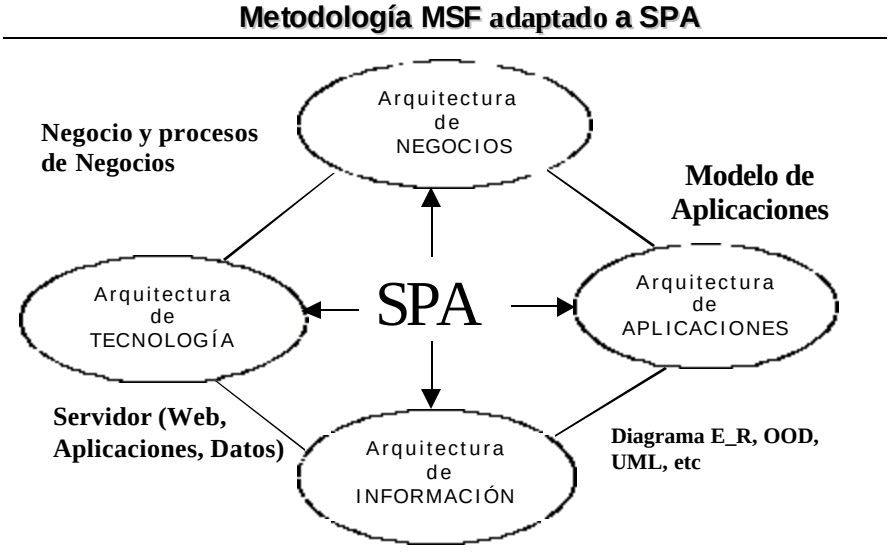


Figura N.º 7. Metodología MSF adaptado a SPA.

Se utilizará la notación **UML** *para el análisis y diseño OO*, **Poseidon for UML 1.6** *como herramienta case* como *herramienta de desarrollo* se usará **Forte for Java 4 Community Edition y** el modelo de la tecnología **Web Services (WS)** [1] para la construcción, funcionalidad e integración de los procesos administrativos de las unidades de negocio con las unidades corporativas de una institución pública.

En la figura N.º 8 se muestra un esquema de interfaces de Web Services con respecto a las capas de cliente, negocios, acceso a datos como una arquitectura del SPA.

el componente que registrar y busca servicios web, por ejemplo tenemos el caso de dos registros UDDI.

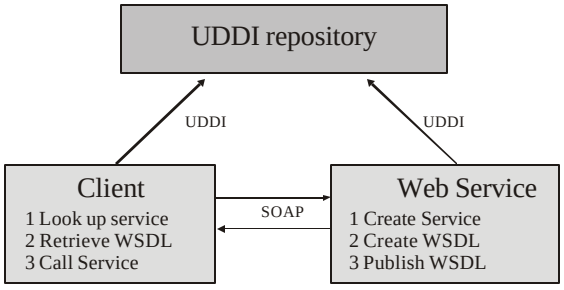


Figura N.º 9. UDDI y sus elementos.

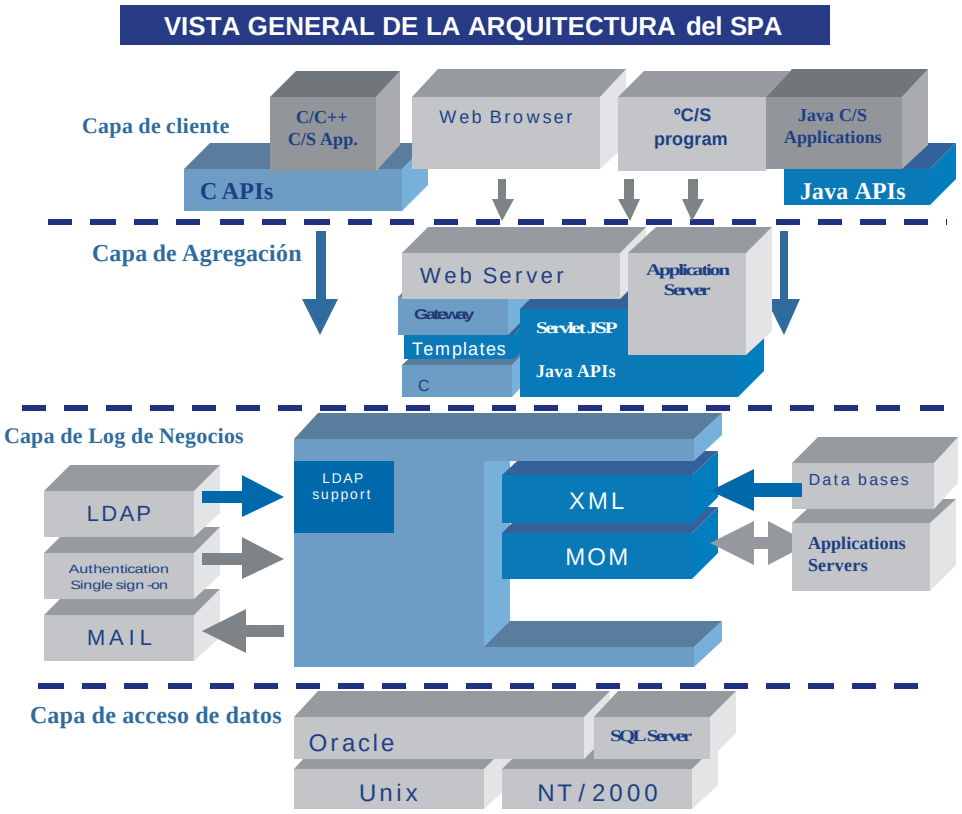


Figura N.º 8. Vista general de la arquitectura del SPA.

Procesos del Modelo Web Services

Los *Web Services* son otra arquitectura de la computación distribuida. Por lo que ,todas las pautas generales para los sistemas del tipo cliente/servidor se aplican también para mejorar la eficiencia de los *Web Services*. En la figura N.º 9 es

Elementos del modelo *Web Services*:

SOAP (Simple Object Access Protocol): Es un protocolo basado en XML para el intercambio de información en un entorno distribuido como se muestra en la figura N.º 10.

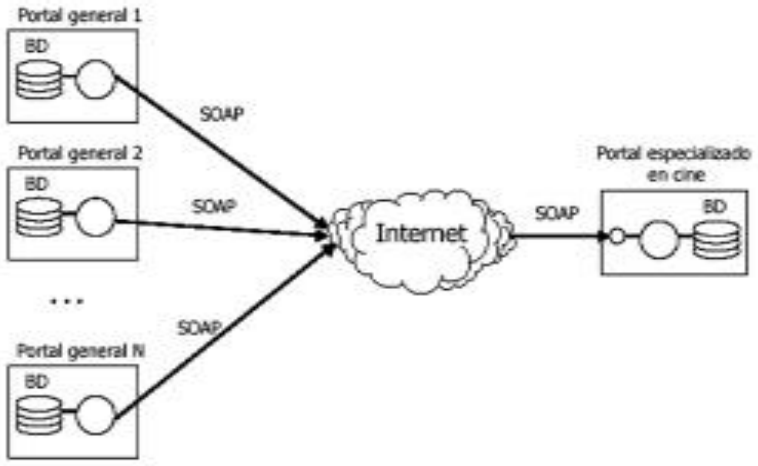


Figura N.º 10. SOAP con BD

UDDI. (Universal, Description, Discovery and Integration): Es una iniciativa de varias empresas (IBM, Microsoft, etc.) que ofrece un servicio gratuito para registrar y buscar servicios web (puede verse en http://www.uddi.org).

En la figura N.º 11 cada servicio web se registra dando, entre otros : (1) su nombre, (2) su(-s) punto(-s) de acceso (ej.: URL) y (3) una descripción del servicio (ej.: la URL de su WSDL, una descripción textual, etc.) De esta manera se convertiría en un servicio de directorio, similar al de las páginas amarillas, de tal forma que una empresa proveedora de servicios web puede publicitarse en UDDI.

WSDL (Web Service Description Language): Este elemento pertenece a la API de programación, en el caso de un lenguaje orientado a objetos, está API permite definir interfaces cuyos métodos se pueden invocar remotamente, en ese sentido es similar a CORBA, es decir WSDL define las interfaces remotas a usar dicha interfaz es un documento XML que define los tipos de datos que usan las operaciones del interfaz (ej.: mediante un esquema XML).

Ejemplo de WSDL:

Tipos de Datos Comunes

```
<s:complexType name=»Player»>
   <s:sequence>
      <s:element minOccurs=»1" maxOccurs=»1"
         name=»Name» nillable=»true»
         type=»s:string» />
      <s:element minOccurs=»1" maxOccurs=»1"
         name=»Average» type=»s:double» />
```

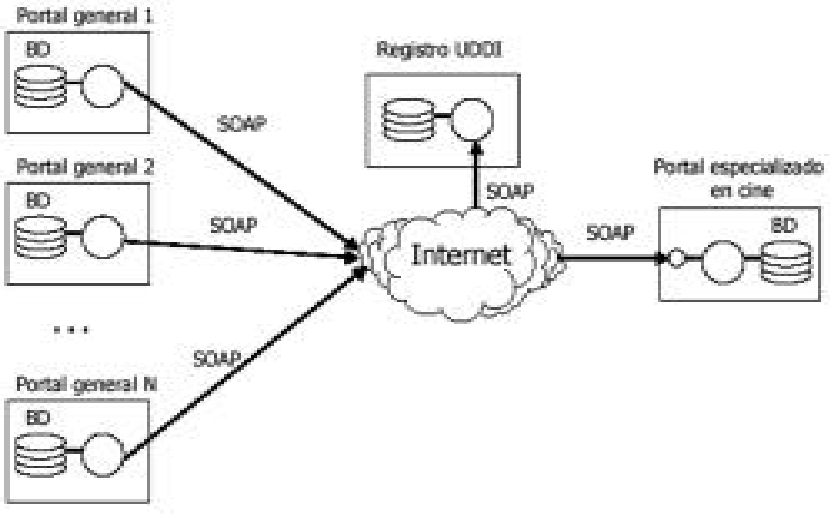


Figura N.º 11. Registro de UDDI

```
        <s:element minOccurs=»1" maxOccurs=»1"
            name=»Year» type=»s:long» />
        <s:element minOccurs=»1" maxOccurs=»1"
            name=»Number» type=»s:short» />
    </s:sequence>
</s:complexType>
```

Formatos de Mensajes

```
<s:element name=»GetBattingAverage»>
    <s:complexType>
        <s:sequence>
            <s:element minOccurs=»1"
            maxOccurs=»1" name=»playerNumber»
            type=»s:short» />
        <s:element minOccurs=»1"
            maxOccurs=»1" name=»year»
            type=»s:long» />
        </s:sequence>
    </s:complexType>
</s:element>
```

Protocolo

```
    <soap:binding
        transport=
        «http://schemas.xmlsoap.org/soap/http»
        style=»document» />
```

Nodos

```
    <soap:address
    location=
    «http://localhost/BaseballService.asmx» />
```

En la Figura N.º 12 se muestra la interfase de *Web Service Cliente* con los elementos UDDI, SOAP, WSDL y el portal esquema que nos permite desarrollar adecuadamente.

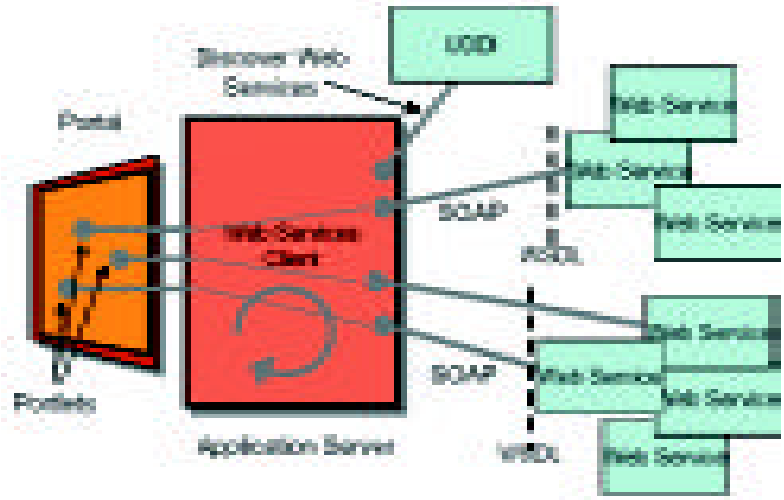
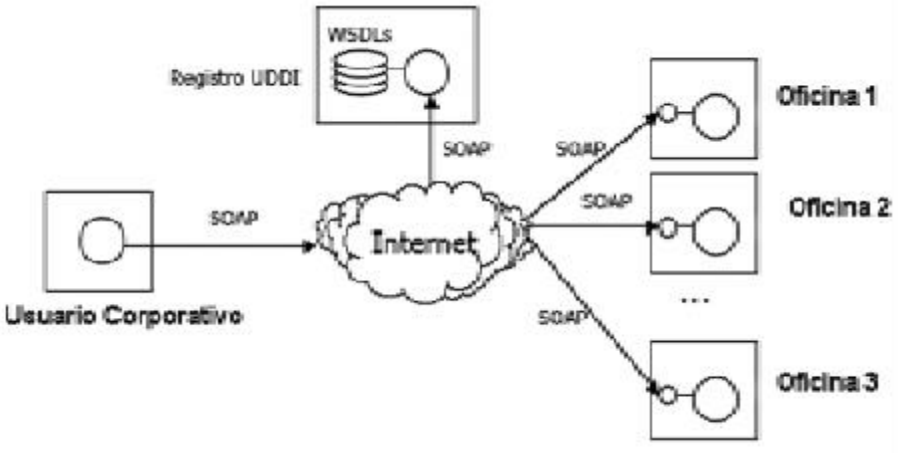
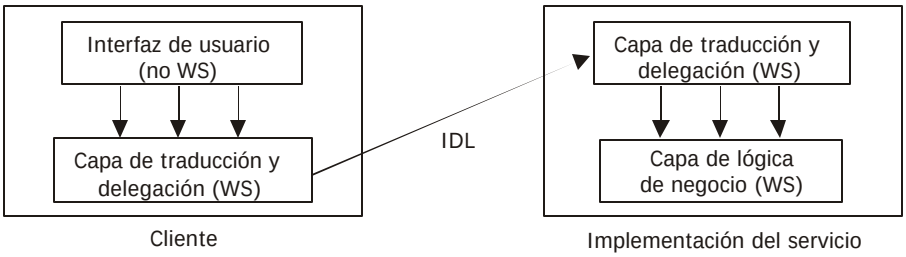


Figura N.º 12. UDDI ,SOAP, WSDL y Web Services

4. Diseño de la aplicación con capas de *Web S ervices*

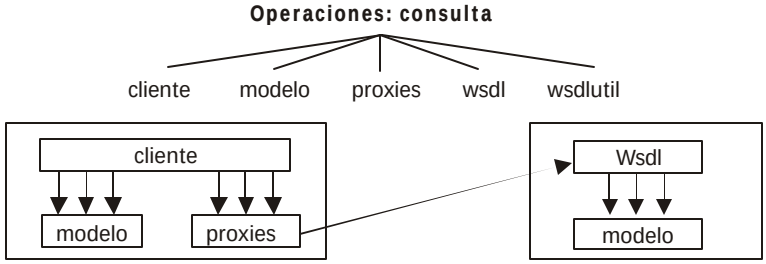


El modelo de implementación:



La mejor manera de implementar esta solución es mediante patrones de diseño por lo que se sugiere la siguiente implementación:

Diagrama de paquetes



Como se aprecia se utilizan los paquetes **modelo** y **proxy.** Posteriormente se indicarán las características de estos patrones.

En la figura N.º 13 mostramos el ***diagrama de clases del paquete modelo*** de acuerdo a lo implementado en Poseidón 1.6 –información.consulta.wsdl.

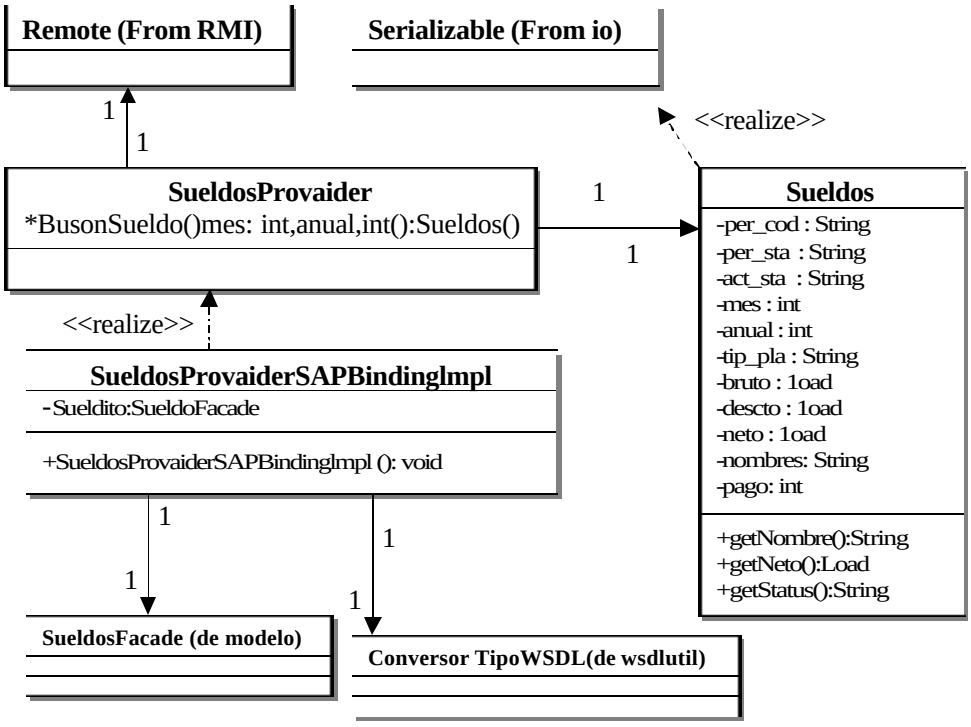


Figura N.º 13. diagrama de clases del paquete modelo.

Obsérvese la diferencia entre los tipos usados en los subpaquetes **model** y **wsdl:**

- En **model** se usa listas de **String** (**java.util.List**) y en **wsdl** se usa **String[]** (tipo Soportado por JAX-RPC).
- En **model** se usa colección de **Sueldos** (**java.util.Collection**) y en **wsdl** se usa **Sueldos**[] (soportado por JAX-RPC).
- En **model** se usa **java.util.Calendar** para las fechas (mejor alternativa que **java.util.Date**), y en **wsdl** se usa **java.util.Date** (no soporta **java.util.Calendar**).

La implementación de esta solución pasa necesariamente, por una forma de desarrollo que se considera un híbrido entre **Corba y WebServices**, dada las características especiales de la organización estudiada.

Considerando de que la comunicación entre cliente y servidor va a ser usando el formato XML, es que se ha definido **SueldosProviderSOAPBindingImpl** la cual es la clase que define al protocolo SOAP el que permitirá el intercambio de mensajes entre ellos. A continuación un fragmento de código de esta clase:

Con respecto al constructor de **SueldosFacade,** este requiere el nombre del directorio que contiene los archivos **.properties,** además para hacer que el código no dependa del nombre del directorio, lo que se hace es leer dicho nombre de la configuración de la aplicación web (**web.xml**) utilizando JNDI (**javax.naming**).

IV. ANÁLISIS DE RESULTADOS

El estudio de SPA ha *determinado que la Tecnología WebServices* es la que presenta una conectividad total entre la *arquitectura propietaria (Microsoft) y las tecnologías en código libre (Java, Apache, etc.)*, el cual desempeñará un papel importante en el desarrollo e integración del servidor de procesos administrativos, *que monitoree y configure la ejecución de tareas a nivel local y el acceso a los usuarios remotos que provengan de un ambiente heterogéneo.*

Fragmento del código de implementación-lógica de negocio distribuido

```
public class SueldosProviderSoapBindingImpl
                              implements SueldosProvider {
  private sueldosFacade SueldosFacade;
  public SueldosProviderSoapBindingImpl() {
    try {
         sueldosFacade = new SueldosFacade(getSueldosDirectoryName());
       } catch (NamingException e) {
                  e.printStackTrace();
                                                    }
  }

   public Sueldos[] findSueldos(Date releaseDate)
                        throws
es.udc.fbellas.corbaws.movies.wsdl.InternalErrorException {
     try {
       Calendar newReleaseDate = Calendar.getInstance();
       newReleaseDate.setTime(releaseDate);
       return CommonWSDLTypeConversor.toWSDL(SueldosFacade.findSueldos(
            newReleaseDate));
        } catch (operaciones..InternalErrorException e) {
             throw CommonWSDLTypeConversor.t oWSDL(e);
          }
   }

   private String getSueldosDirectoryName() throws NamingException {
     InitialContext initialContext = new InitialContext();
     return (String) initialContext.lookup("java:comp/env/"+"SueldosDirectoryName");
    }
}
```

Se ha determinado los indicadores estándares de eficiencia en la implantación de cualquier SPA para instituciones publicas como la UNMSM y éstos son: independencia total en el diseño del cliente y, el servidor, comunicación a través de internet / intranet/extranet de clientes y servidores, eficiencia en el mantenimiento de la tecnología existente, menor costo de implementación, menor complejidad, facilidad y flexibilidad en la adición de nuevos clientes, libre elección de la plataforma de desarrollo.

Se ha definido el hardware y software que permita la comunicación entre unidades funcionales de UNMSM y poder ofrecer a los usuarios corporativos (Rector, Vicerrectores, etc.) acceso a información particular de cada oficina y además a información que proviene de la mezcla de varias oficinas, en forma transparente y utilizando el menor costo posible.

Se ha diseñado la aplicación con capa *Web Service* para una unidad de negocio(Oficina de Sueldos de la UNMSM) con la Unidad Corporativa (RECTORADO) considerando de que la comunicación entre cliente y servidor va a ser usando el formato XML.

V. CONCLUSIONES

1. Los procesos administrativos de cualquier institución, siempre son el cuello de botella, para una buena gestión, mas aun cuando se trata de redes corporativas.
2. El servidor de procesos administrativos, es una herramienta para el control y acceso rápido de los recursos y actividades de cualquier organización, para una toma eficiente de decisiones.
3. Las tecnologías existentes para una implementación de sistemas heterogéneos vistas en este informe técnico se desprende que la tecnología mas moderna y llamada a ser la que permitirá la *conectividad total entre la arquitectura propietaria (Microsoft) y las tecnologías en código libre (Java, Apache, etc.)* será ***Web Services*** el cual desempeñará un papel importante en la integración que este proyecto de investigación esta fundamentando.
4. Es posible implementar un servidor de procesos administrativos, que permita la comunicación entre las unidades funcionales y corporativas de una organización, en forma transparente empleando el menor costo posible.
5. Los servicios web, permitirán que se sigan usando los sistemas existentes en las empresas y que éstas se conecten a la vez con sus socios de negocios, haciendo uso de internet como canal de comunicación. Esto no excluye la posibilidad de crear servicios web al interior de las organizaciones (intranet) y con algunos aliados particulares (extranet), para lograr así la optimización de procesos y operaciones administrativos en una organización pública.

Agradecimiento

El responsable del proyecto agradece a los profesores: Luzmila Pro, Carlos Yañez, Augusto Cortez y Jimy Espezua, por sus sugerencias en la elaboración del presente artículo.

VI. BIBLIOGRAFÍA

1. Mike Clark., Peter Fletcher., J. Jeffrey Hanson. *Web services business strategies and architectures*. McGraw - Hill, (1999).
2. Eric Newcomer. *Understanding Web Services: XML*, WSDL, SOAP, and UDDI. Addison-Wesley, (2000).
3. Chow, Randi and Johnson. *Distributed operating systems and algorithms*. McGraw Hill, (1998).
4. Marilee Ford., Kim Lew., Steve Spannier. *Tecnologías Java Enterprise para aplicaciones web*. Prentice-Hall, Cisco Press (1999).
5. Martin, James. *Computer networks and distributed processing: software, techniques and architecture.* McGraw Hill, (1997).
6. Mike Clark., Peter Fletcher., Jeffrey Hanson, *Web Services Business Strategies and Architectures Journal ACM.* (2002).
7. GDB/RBD. *MPI primer Developing with LAM*, The Ohio State University, Japon (1999).
8. Ramesh Nagappan., Robert Skoczylas. *Rima Patel Sriganesh, Developing Java Web Services: Architecting and Developing Secure Web Services Using Java*. Journal IBM (2002).