Revista Iberoamericana de Producción Académica y Gestión Educativa
ISSN: 2007 - 8412

Algoritmos estratégicos para analizar código WSDL de diversas plataformas

Strategic Algorithms to Analize WSDL Code from Diverse Platforms

Elías Rivera Custodio
Universidad Tecnológica del Usumacinta, México
elias300481@hotmail.com

René Santaolaya Salgado
Tecnológico Nacional de México**,** Centro Nacional de Investigación y Desarrollo Tecnológico, México
rene@cenidet.edu.mx

Olivia G. Fragoso Díaz
Tecnológico Nacional de México**,** Centro Nacional de Investigación y Desarrollo Tecnológico, México
fragoso@cenidet.edu.mx

Martín Gerardo Martínez Valdés
Universidad Tecnológica del Usumacinta, México
martingerardo16@hotmail.com

Resumen

El objetivo de este trabajo de investigación fue diseñar e implementar en un programa de cómputo una familia de algoritmos que fuera capaz de analizar el código de lenguaje de descripción de servicios web (WSDL, por sus siglas en inglés) de diversas plataformas. El documento WSDL describe las credenciales de un servicio web, especificando datos como el nombre del servicio, los métodos que ofrece, los parámetros de entrada y el tipo de dato retornado. También se puede encontrar información referente a su dirección y sus puertos de entrada y salida. Toda esta información es utilizada por las aplicaciones clientes que

requieren consumir un servicio web. Aunque el documento WSDL es un estándar, las diferentes plataformas de desarrollo generan documentos diferentes, lo que evita que una plataforma pueda leer el WSDL generado en otra. Para solventar la problemática, la idea fue integrar una familia de algoritmos para analizar documentos de diferentes fuentes, y construir una arquitectura de clases flexible, en el sentido de que se puedan integrar nuevos algoritmos sin necesidad de reconstruir la arquitectura original. Esto se logró gracias a la implementación del patrón de diseño *Strategy*, el cual permite integrar una serie de algoritmos que realizan la misma función pero de forma diferente (Gamma, Helm, Johson y Vlissides, 1995). Como resultado, se obtuvo un programa de cómputo que analiza documentos creados con las plataformas Axis y NetBeans, así como una arquitectura de clases capaz de añadir nuevos algoritmos de análisis.

Palabras clave: patrón de diseño, programa de cómputo, servicio web, *Strategy*, WSDL.

Abstract

The objective of this research was to design and implement in a computer program an algorithm family to be able to analyze the Web Services Language Description (WSDL) code from diverse platforms. The WSDL describes the web service credentials, specifying data such as the name of the service, the methods that offer and the incoming and returning data; can also be found information about service address and in and out ports. This information is used by the client applications that require consuming a web service. Although the WSDL document is a standard, the different platforms of developing services generate different documents, that avoid that a platform could read the WSDL generated in another one. To solve this problematic, the idea was integrating an algorithm family to analyze documents from different sources, and to build a flexible class architecture, in the sense to integrate new algorithms without the necessity to rebuild the original architecture, this was gotten thanks to the implementation of the Strategy design, that allows to integrate an algorithm series that do the same function but in a different way (Erich, 1995). As a result, a computer program was obtained to analyze created documents by the Axis and NetBeans, moreover of a class architecture able to integrate new analysis algorithms.

Keywords: design pattern, computer program, web service, Strategy, WSDL.

Revista Iberoamericana de Producción Académica y Gestión Educativa
ISSN: 2007 - 8412

Fecha Recepción: Febrero 2018 **Fecha Aceptación:** Junio 2018

Introducción

Los servicios web son aplicaciones lógicas programables que permiten el intercambio de datos entre aplicaciones vía Internet. Y tienen una interfaz descrita en un documento llamado *lenguaje de descripción de servicios web* (WSDL, por sus siglas en inglés) (World Wide Web Consortium [W3C], 2004). Utilizando esta tecnología, una organización puede brindar servicios, intercambiar información, completar sus transacciones y conseguir una completa integración de sus procesos de negocios.

Actualmente el desarrollo de software está basado en la programación modular. Dicha ideología es utilizada en la implementación de servicios web. Cuando uno de ellos no cubre por completo los requerimientos de una aplicación cliente, una solución es la composición, es decir, crear un servicio a partir de otros y con esto lograr que este nuevo servicio brinde toda la funcionalidad requerida por la aplicación cliente. La interacción de varios servicios web da origen al concepto composición de servicios (Orozco, 2009).

En el Centro Nacional de Investigación y Desarrollo Tecnológico (CENIDET) se han desarrollado diversas herramientas de composición de servicios web que, desde su concepción, contemplaron la creación de módulos como el analizador de documentos WSDL —donde, además, se implementó un algoritmo para analizar dicho documento. El documento WSDL describe por completo la interfaz de comunicación con el servicio web (Guzmán, 2006).

Como producto de los trabajos desarrollados en el CENIDET referentes a la composición se mencionan los siguientes.

Composición de servicios web

El objetivo general de esta tesis fue el desarrollo de una secuencia de pasos soportados por una herramienta que permitiera al desarrollador de *software* crear aplicaciones informáticas específicas, mediante la composición de servicios web, reduciendo tiempo, costo y complejidad de producción (Guzmán, 2006).

Composición de servicios web utilizando diagramas de actividad

En este trabajo se implementó un método que permite construir nuevas aplicaciones a partir de la composición de servicios web disponibles en Internet. La composición se modela a través de diagramas de actividad de lenguaje unificado de modelado (UML) (Fowler y Scott, 1999), que posteriormente son traducidos a código de lenguaje de ejecución de proceso de negocio (BPEL), el cual representa el servicio compuesto. Para ejecutar el código BPEL se utiliza una máquina de ejecución de Oracle (De Gyvés, 2007). Los diagramas de actividad UML permiten mostrar un proceso de negocio o un proceso de *software* como un flujo de trabajo a través de una serie de acciones (Microsoft, 2015), utilizando símbolos gráficos para su diseño.

Extensión al sistema de composición de servicios web utilizando diagramas de actividad

En este proyecto de investigación se buscó generar de manera automática código en lenguaje de ejecución de procesos de negocio con servicios web (BPEL4WS, por sus siglas en inglés) que tenga como entrada un diagrama de actividad y que tenga como salida código en BPEL4WS semánticamente equivalente al diagrama. En particular, se incorporaron los elementos *fork*, *join*, mensajes de flujo, expresiones de resguardo y *note* del diagrama de actividad integrados en la interfaz del sistema WS-SIDDOO (Orozco, 2009). En la actualidad BPEL4WS se ha convertido en un estándar para la composición de servicios web; permite invocar servicios, manipular datos, comunicar fallas y finalizar procesos (IBM, 2017).

Estas herramientas tienen en común el módulo analizador de documentos WSDL, el cual puede leer solamente documentos de servicios web creados con la plataforma JDeveloper, lo que implica que son incapaces de analizar y, por ende, de componer servicios creados en otras plataformas, tales como Axis, NetBeans o Systinet.

El objetivo principal de esta investigación fue crear un programa de cómputo capaz de analizar documentos WSDL de diversas plataformas, con el fin de ampliar las posibilidades de composición de servicios web concebidos con tecnologías de desarrollo diferentes.

Metodología

Análisis del módulo analizador de documento WSDL

El análisis se basó en la arquitectura de clases con el fin de estructurar una arquitectura de clases flexible, capaz de integrar nuevos algoritmos sin necesidad de reestructurar nuevamente el sistema.

Análisis del documento WSDL de cada plataforma

Este análisis es importante porque determina la lógica que debe emplear el algoritmo para extraer la información requerida por la aplicación cliente. La información que se debe extraer del documento es la siguiente: nombre del servicio, dirección del servicio, métodos que ofrece, parámetros de entrada (tipos de datos), tipo de dato retornado y puertos de entrada y salida.

En el caso de las plataformas Axis y JDeveloper, el documento WSDL especifica toda la información del servicio; en NetBeans es diferente debido a que la descripción del servicio se presenta en dos fuentes: el documento WSDL y el documento llamado *Definición de esquema XML* (XSD, por sus iniciales en inglés) (W3C, 2015). Es en este último donde se especifica toda la información detallada del servicio web. En este caso, el documento XSD representa una extensión del WSDL (W3C, 2007).

A continuación se presenta un análisis del documento XSD generado por la plataforma NetBeans. En el siguiente fragmento de código se identifican tres métodos: “multiplica”, “suma” y “vec”:

```
<xs:element name="multiplica"
        type="tns:multiplica" />
  <xs:element name="multiplicaResponse"
        type="tns:multiplicaResponse" />
  <xs:element name="suma"
        type="tns:suma" />
  <xs:element name="sumaResponse"
        type="tns:sumaResponse" />
```

```
<xs:element name="vec"
        type="tns:vec" />
<xs:element name="vecResponse"
        type="tns:vecResponse" />
```

En el siguiente fragmento, por su parte, se especifican los parámetros recibidos por los métodos “multiplica”, “suma” y “vec”. También se especifica el tipo de cada parámetro y el tipo de dato retornado por cada método. Por ejemplo, el primer método se llama *multiplica*, recibe dos parámetros de tipo entero (“a” y “b”) y retorna un tipo de dato entero.

```
<xs:complexType name="multiplica">
        <xs:sequence>
                <xs:element name="a"
                        type="xs:int" />
                <xs:element name="b"
                        type="xs:int" />
        </xs:sequence>
</xs:complexType>
<xs:complexType name="multiplicaResponse">
        <xs:sequence>
                <xs:element name="return"
                        type="xs:int"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="suma">
        <xs:sequence>
                <xs:element name="x"
                        type="xs:int" />
                <xs:element name="y"
                        type="xs:int" />
```

```
        </xs:sequence>
</xs:complexType>
<xs:complexType name="sumaResponse">
        <xs:sequence>
                <xs:element name="return"
                        type="xs:int"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="vec">
        <xs:sequence>
                <xs:element name="a"
                        type="xs:int" />
                <xs:element name="b"
                        type="xs:int" />
                <xs:element name="c"
                        type="xs:int" />
        </xs:sequence>
</xs:complexType>
<xs:complexType name="vecResponse">
        <xs:sequence>
                <xs:element name="return"
                        type="xs:anyType"/>
  </xs:sequence>
</xs:complexType>
```

Realizando la traducción del documento XSD en un lenguaje de programación como Java, el código del servicio web sería el siguiente:

```
int multiplica(int a, int b)
        {//código          }
int suma(int x, int y)
        {//código          }
void vec(int a, int b, int c)
        {//código          }
```

Implementación

La implementación se realizó utilizando el lenguaje de programación Java con el entorno de desarrollo integrado (IDE) MyEclipse. Para el análisis del documento WSDL y XSD se utilizó la API JDOM (JDOM, 2015), la cual permite interpretar el documento como una estructura de árbol.

A continuación, se muestra la estructura del programa:

```
   //Estructura de la clase Servicio
class servicio
{           //inicio de la clase
 private String direccion;
 private String nombre;
 private String puerto;
 private String plataforma;
 private String[][] metodos=new String[4][100];

   String getDireccion() //Obtiene la dirección del servicio
  {  }
  String getNombre()   //Obtiene el nombre del servicio
  {  }
  String getPuerto()     //Obtiene los puertos de entrada y salida del servicio
  {  }
 String getPlataforma(String URL)  //Obtiene la plataforma de análisis
```

```
  {   }
      // Este método se encarga de guardar en una matriz
     //los métodos, y tipos de datos.
  void setMetodos(int x, int y,String parametro)
  {   }
  String[][] getMetodos()
  {   }
}              //fin de la clase Servicio

    //Estructura de la clase AnalizadorWSDL
abstract class AnalizadorWSDL
{ protected servicio ser;
  abstract void analiza(String URL);
}

    //Clase derivada Analizador_NetBeans
class Analizador_NetBeans extends AnalizadorWSDL
{
  public Analizador_NetBeans(servicio servi)
     {  ser=servi;  }
  public void analiza(String URL)
     {  }
 public void analizaXSD(String URLXSD)
     {  }
}
```

Pruebas

La primera prueba se realizó tomando el XSD mostrado como ejemplo en la fase dos de la sección Metodología. Este XSD describe un servicio que implementa tres métodos ya mencionadas anteriormente: “multiplica”, “sumar” y “vec”.

La salida del programa al analizar el XSD fue la siguiente:

Método multiplica int a
Método multiplica int b
Tipo de dato retornado multiplicaResponse int return
Método suma int x
Método suma int y
Tipo de dato retornado sumaResponse int return
Método vec int a
Método vec int b
Método vec int c
Tipo de dato retornado vecResponse anyType return

En otra prueba se utilizó un XSD que especifica un método llamado *clasifica_animal*. En este documento se describe un tipo de dato abstracto denominado *animal*:

```
<xs:element type="tns:clasifica_animal" name="clasifica_animal"/>
<xs:element type="tns:clasifica_animalResponse" name="clasifica_animalResponse"/>-
<xs:complexType name="clasifica_animal">
    <xs:sequence>
        <xs:element type="tns:animal" name="linked_animales"
        maxOccurs="unbounded" minOccurs="0"/>
        <xs:element type="xs:string" name="tipo" minOccurs="0"/>
    </xs:sequence>
</xs:complexType>
<xs:complexType name="animal">
    <xs:sequence>
        <xs:element type="xs:string" name="nombre" minOccurs="0"/>
        <xs:element type="xs:string" name="tipo" minOccurs="0"/>
        <xs:element type="xs:float" name="peso_maximo"/>
```

```
                <xs:element type="xs:float" name="peso_minimo"/>
                <xs:element type="xs:float" name="velocidad_maxima"/>
                <xs:element type="xs:float" name="edad_promedio"/>
                <xs:element type="xs:string" name="regiones_habitables" minOccurs="0"/>
                <xs:element type="xs:int" name="existencia"/>
        </xs:sequence>
</xs:complexType>
<xs:complexType name="clasifica_animalResponse">
        <xs:sequence>
                <xs:element type="tns:animal" name="return" maxOccurs="unbounded"
                minOccurs="0"/>
        </xs:sequence>
</xs:complexType
```

A continuación se muestra la salida del programa:

```
Método clasifica_animal animal linked_animales
Método claseifica_animal string tipo
Objeto animal string nombre
Objeto animal string tipo
Objeto animal float peso_maximo
Objeto animal float peso_minimo
Objeto animal float velocidad_maxima
Objeto animal float edad_promedio
Objeto animal string regiones_habitables
Objeto animal int existencia
Tipo de dato retornado calsifica_animalResponse animal return
```

El analizador tiene la capacidad de extraer datos de estructuras abstractas, no solo datos primitivos.

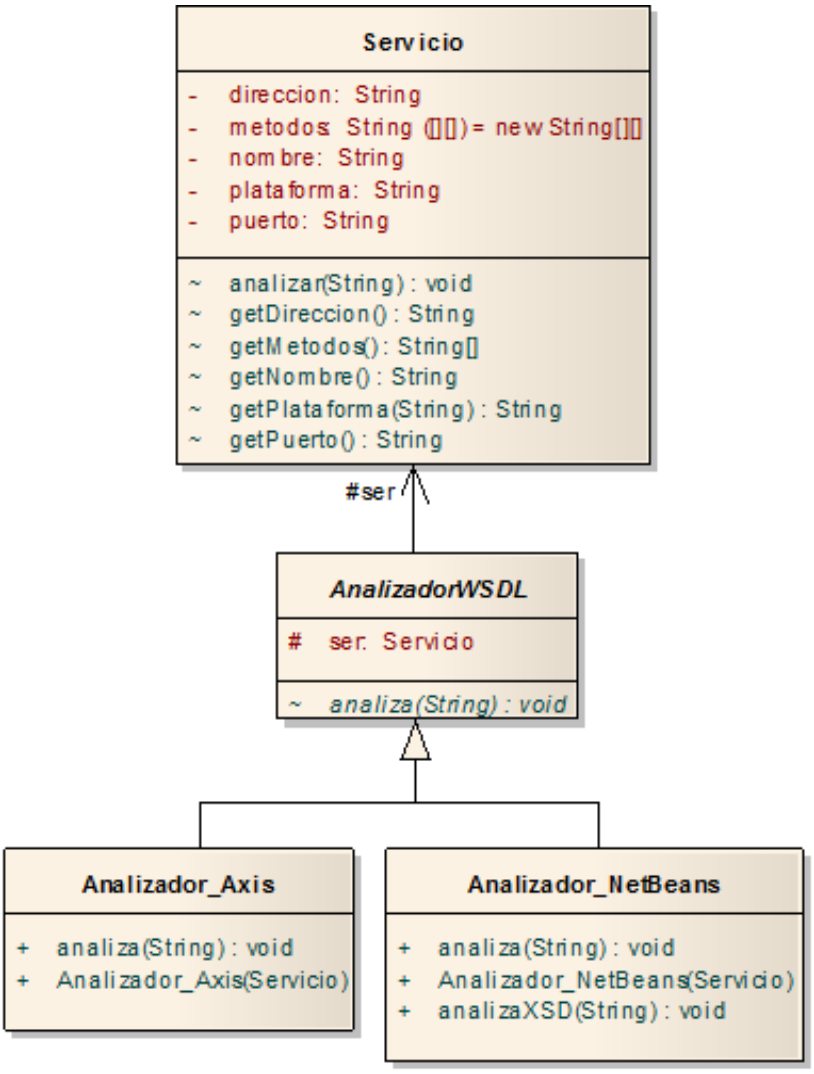
Resultados

Como resultado se tiene un programa de cómputo capaz de leer documentos WSDL de las plataformas NetBeans y Axis. También se cuenta con una arquitectura de clases flexible, basada en el patrón de diseño *Strategy*, lo cual permitirá integrar nuevos algoritmos sin necesidad de reestructurar el programa. La finalidad de este patrón de diseño es agrupar algoritmos que realizan la misma función pero de forma diferente.

En la figura 1 se muestra la arquitectura de clases del programa, en donde la clase "Servicio" contiene los métodos y atributos necesarios para extraer y almacenar los datos del XSD. Por ejemplo, el atributo "métodos" es un arreglo que sirve para almacenar todos los métodos que ofrece el servicio web, incluyendo los parámetros de entrada y salida con sus respectivos tipos de datos. La clase "AnalizadorWSDL" es de tipo abstracto y brinda la posibilidad de encapsular una serie de algoritmos en clases que deriven de ella.

El método "analiza()" de la clase "AnalizadorWSDL" es de tipo abstracto y es implementado en las clases derivadas. Cada una de las clases analizadoras, al ser instanciadas, recibirá como parámetro un objeto de tipo "Servicio" para almacenar en él los resultados del análisis.

Figura 1. Arquitectura de clases del sistema



Fuente: Elaboración propia

Primero se desarrolló el analizador para documentos generados por la plataforma Axis, posteriormente se anexó el analizador para documentos generados con NetBeans. Esta acción no implicó reestructurar el programa. La clase "Analizador" para la plataforma NetBeans incluye un método llamado *analizaXSD* debido a que esta plataforma, además del WSDL, también genera un documento XSD, como ya se mencionó anteriormente.

Conclusiones

La heterogeneidad de los documentos WSDL ha estado en cuestión debido a que al tratarse de un estándar no debería existir diferencia entre los documentos generados por las diferentes plataformas; y sin embargo, existe. En la actualidad se cuenta con diversas herramientas de desarrollo de servicios web y cada una crea WSDL con ciertas particularidades, lo que dificulta crear una aplicación que sea capaz de analizar documentos creados en diferentes entornos.

Este trabajo brinda una solución a esta problemática por medio de una herramienta que puede analizar diferentes WSDL independientemente de la plataforma en que fueron creados. Esto se logró mediante la implementación del patrón de diseño *Strategy*, el cual permite implementar tantos algoritmos como analizadores que se quieran desarrollar. Con esta arquitectura se evita que la inclusión de nuevos algoritmos implique la reestructuración de la aplicación.

Se propone anexar a las herramientas de composición desarrolladas en el CENIDET este programa para brindarles mayor soporte y así puedan realizar la composición de servicios desarrollados en plataformas diferentes a JDeveloper. También se plantea el desarrollo de nuevos algoritmos de análisis e incluirlos a la herramienta desarrollada en este proyecto.

Referencias

De Gyvés, A. (2007). *Composición de servicios web utilizando diagramas de actividad* (tesis de maestría). Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México.

Fowler, M. y Scott, K. (1999). *UML gota a gota*. México: Pearson educación.

Gamma, E., Helm, R., Johnson R. y Vlissides, J. (2003). *Patrones de diseño. Elementos de software orientado a objetos reutilizables*. Madrid, España: Pearson educación.

Guzmán, M. (2006). *Composición de servicios web* (tesis de maestría). Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México.

IBM. (2017). Business Process with BPEL4WS. Recuperado de https://www.ibm.com/developerworks/library/ws-bpelcol1/.

JDOM. (2015). JDOM v2.0.6 API Specification. Recuperado de http://www.jdom.org/docs/apidocs/.

Orozco, M. (2009). *Extensión al sistema de composición de servicios web utilizando diagramas de actividad* (tesis de maestría). Centro Nacional de Investigación y Desarrollo Tecnológico, Cuernavaca, Morelos, México.

Microsoft. (2015). Diagramas de actividades UML: Referencia. Recuperado de https://msdn.microsoft.com/es-es/library/dd409360.aspx.

World Wide Web Consortium [W3C] (R). (2004). Web Services Architecture. Recuperado de https://www.w3.org/TR/ws-arch/#whatis.

World Wide Web Consortium [W3C] (R). (2007). Semantic Annotations for WSDL and XML Schema. Recuperado de https://www.w3.org/TR/sawsdl/.

World Wide Web Consortium [W3C] (R). (2015). XML Technology. Recuperado de https://www.w3.org/standards/xml/schema.