

Web Services.

Introducción y Escenarios para su Uso.

Moisés Daniel Díaz Toledano. www.moisesdaniel.com

- 1.- [Introducción a los WebServices.](#)
 - 1.1.- [Definición de la W3C.](#)
 - 1.2.- [Un poco de perspectiva.](#)
 - 1.3.- [Qué son WSDL, SOAP y HTTP.](#)
 - 2.- [Escenarios de uso de los WebServices.](#)
 - 2.1.- [Qué es la EAI \(Enterprise Application Integration\).](#)
 - 2.2.- [Plataformas EAI y WebServices.](#)
 - 2.3.- [Qué es SOA \(Service Oriented Architecture\).](#)
-

1.- Introducción a los WebServices.

Si buscamos los términos “Web Service” en Google nos salen como resultado más de 13 millones de páginas.

Evidentemente no todas hablan sobre el tema, pero puede decirse que los servicios Webs son una de las tecnologías más atractivas del mundo internet.

Hace ya algún tiempo que esta tecnología dejó de ser una moda y una promesa, para convertirse en una realidad con muchas utilidades. Como muestra volvamos a Google. Esta compañía inició en Abril del año 2002 la posibilidad de hacer búsquedas usando su motor de búsqueda (que indexa más de 3.000 millones de documentos Web). Esto se hace mediante webservices. Como podemos ver, una aplicación real para el mundo real.

Pero, ¿qué es un webservice?

1.1.- Definición de la W3C.

Según la W3C (el organismo que se encarga de desarrollar gran parte de los estándares de internet), se define un webservice de la siguiente forma: “Un servicio Web es una aplicación software identificada mediante una URI, cuyo interfaz (y uso) es capaz de ser definido, descrito y descubierto mediante artefactos XML, y soportar interacciones directas con otras aplicaciones software usando mensajes basados en XML y protocolos basados en Internet”.

Qué, ¿cómo os habéis quedado? Además de ser una definición un tanto complicada, uno llega a la conclusión de que es tan genérica que millones de cosas pueden ser un Webservice.

Sin embargo, cuando los desarrolladores hablamos de Web Services nos estamos refiriendo a tecnologías muy concretas, al menos la gran mayoría de las veces.

Por todo esto una definición alternativa podría ser la siguiente: Un servicio Web es un componente software que se basa en las siguientes tecnologías:

- ✍ Un formato que describa la interfaz del componente (sus métodos y atributos) basado en XML. Por lo general este formato es el WSDL (Web Service Description Language).
- ✍ Un protocolo de aplicación basado en mensajes y que permite que una aplicación interactúe (use, instancia, llame, ejecute) al webservice. Por lo general este protocolo es SOAP (Simple Object Access Protocol).
- ✍ Un protocolo de transporte que se encargue de transportar los mensajes por internet. Por lo general este protocolo de transporte es HTTP (Hiper-Text Transport Protocol) que es

exactamente el mismo que usamos para navegar por la Web.

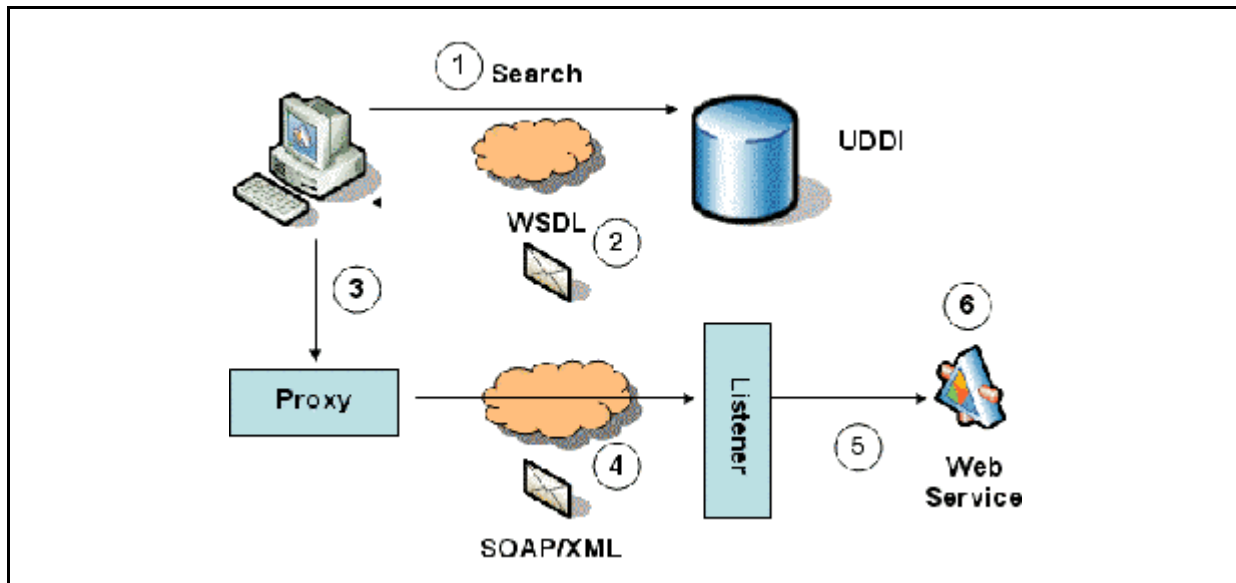


Figura 1. Proceso y tecnologías de los Servicios Web. [[Figuras tomadas 'around the Web']]

Los servicios Web, no son por tanto aplicaciones con una interfaz gráfica con la que las personas puedan interactuar, sino que son software accesible en internet (o en redes privadas que usen tecnologías internet) por otras aplicaciones. De esta forma podemos desarrollar aplicaciones que hagan uso de otras aplicaciones que estén disponibles en internet interactuando con ellas.

Un típico ejemplo podría ser un WebService al que se le pudiese preguntar por una empresa y que nos retornase en tiempo real el valor al que están cotizando las acciones de dicha compañía.

De esta forma cualquier aplicación (ya sea web o de escritorio) que quiera mostrar esta información sólo tendría que solicitarla a través de internet al servicio Web cuando la necesitase.

Otro ejemplo de servicio Web podría ser uno que al pasarle el nombre de una ciudad, nos devolviese la temperatura, humedad, y otras condiciones climatológicas de la misma.

Los servicios Web no son la panacea, sino una tecnología apropiada para resolver ciertos problemas.

Básicamente los servicios Web permiten que diferentes aplicaciones, realizadas con diferentes tecnologías, y ejecutándose en toda una variedad de entornos, puedan comunicarse e integrarse, lo cual es muy importante, y ¿por qué? ... tomemos un poco de perspectiva:

1.2.- Un poco de Perspectiva.

El coste de desarrollar software siempre ha sido altísimo, y la diversidad de las plataformas una realidad desde el inicio de la informática. De hecho conforme más complejas fueron las aplicaciones que las empresas demandaban, más caras era desarrollarlas. Para enfrentarse a

esta situación se han seguido diferentes líneas todas ellas encaminadas a reutilizar las aplicaciones ya desarrolladas.

Una de estas propuestas fue (y es) la estandarización de lenguajes de programación de tal forma que si cualquiera de nosotros escribía un programa en C, solo necesitaríamos un compilador de C en la plataforma específica en la que quisiéramos ejecutar la aplicación. Esto en la realidad ha sido (y es) difícil de hacer funcionar, porque en seguida surgieron pequeñas diferencias y extensiones de C que hacían difícil ‘transportar’ una aplicación entre diferentes plataformas.

Otra posibilidad ha sido la que ha desarrollado Sun con Java. Se programa para una plataforma, pero para una plataforma virtual, en este caso para la máquina virtual Java, y en cada plataforma real se implementa una máquina virtual de java, que será la encargada de ejecutar las aplicaciones escritas en Java. Las implementaciones son específicas de cada plataforma pero al ser todas las máquinas virtuales exactamente la misma, los programas escritos en Java deben poder ejecutarse sin ningún problema en todas las plataformas que tengan una máquina virtual de Java. Esta apuesta ha demostrado ser muy útil y funcionar bastante bien.

Sin embargo, ¿qué pasa si tenemos varias aplicaciones ya desarrolladas en lenguajes propietarios o en plataformas específicas y queremos que interaccionen entre ellas? El coste de elegir a posteriori un único lenguaje o plataforma y migrarlo al mismo es descabellado en la mayoría de las situaciones, y es aquí donde los WebServices, así como otras tecnologías, pueden sernos de una grandísima utilidad.

Con los WebServices podemos reutilizar desarrollos ya utilizados sin importar la plataforma en la que funcionan o el lenguaje en el que están escritos. Los servicios Web se constituyen en una capa adicional a estas aplicaciones de tal forma que pueden interaccionar entre ellas usando para comunicarse tecnologías estándares que han sido desarrolladas en el contexto de internet.

1.3.- Qué son WSLD, SOAP y HTTP.

Los estándares son definiciones o formatos que se aprueban o reconocen desde organizaciones de estandarización. Generalmente estos organismos están formados por el conjunto de empresas más representativas de un sector o de un campo de la producción.

Los estándares permiten que las industrias desarrollen componentes con las garantías suficientes de: interacción, funcionalidad y calidad,

Ayudan a desarrollar los bloques básicos sobre los que seguir construyendo el edificio tecnológico.

Los estándares son extremadamente importantes en la informática, ya que permiten que se combinen productos de diferentes fabricantes para el desarrollo de sistemas, tanto software como hardware.

Sin estándares, sólo los productos de la misma compañía podrían ser usados de forma conjunta.

Actualmente existen estándares para diversos protocolos de comunicación, formatos de datos, lenguajes de programación, etc.

Los organismos más importantes de estandarización son: ANSI, IEEE, ISO, y W3C.

Los webservices se construyen sobre estándares y a su vez pretenden ser un estándar con los que construir sistemas a partir de piezas dispares, desarrollas por distintos fabricantes,

funcionando en distintos sistemas, y construidas con distintas tecnologías.

Los principales estándares para el desarrollo de Webservices son los siguientes:

☞ **SOAP** es el acrónimo de Simple Object Access Protocol, es decir protocolo simple de acceso a objetos. SOAP es un protocolo ligero de mensajes XML que se usa para codificar la información de los mensajes de petición y respuesta de los WebServices que se envían a través de una red.

Los mensajes SOAP son independientes de los sistemas operativos y de los protocolos, y pueden ser transportados usando una variedad de protocolos internet, incluyendo SMTP, y HTTP.

Expliquémoslo un poquito mejor. Dentro del paradigma orientado a objetos, usar un Webservice es igual que usar cualquier otra clase. Y esto significa instanciarlo, y llamar a sus métodos, pasándoles los parámetros que sean necesarios, y obteniendo a su vez el resultado que nos retornen.

Como ya hemos dicho por lo general llamaremos a WebServices que no estarán en nuestra máquina local, sino en cualquier servidor accesible desde internet. Debemos por tanto de disponer de alguna forma de llamar a cualquiera de sus métodos pasándole los parámetros oportunos y obteniendo el resultado de esa llamada (si es que el método devuelve algo después de ser ejecutado).

Soap es un protocolo que define precisamente cómo realizar esta comunicación, es decir cómo debemos codificar las llamadas a los métodos de un webservice, y cómo debe el webservice codificar el resultado para que nosotros lo podamos interpretar.

Estos mensajes son los que transportarán los protocolos de transporte, por lo general, HTTP.

☞ **WSDL** es un acrónimo de Lenguaje de Descripción de Servicios Web (Web Services Description Language), que es un lenguaje XML usado para describir la interfaz de un webservice como un conjunto de puntos finales de comunicación (métodos) capaces de intercambiar mensajes (es decir recibir llamadas con sus parámetros correspondientes y generar respuesta con el resultado que le corresponda). WSDL se considera parte integral de UDDI, que debería ser un registro de servicios Web XML (esto último lo explicamos un poquito más abajo).

WSDL es el lenguaje usado por UDDI para describir a los webservices. Fue desarrollado de forma conjunta por Microsoft e IBM.

De ejemplo, generemos el fichero WSDL de una pequeña clase:

```
package pruebas;

public class calc {
    public calc() {
    }

    public int suma(int a, int b){
        return a + b;
    }
}
```

Usando como generador Oracle JDeveloper, su WSDL sería:

```
<?xml version = '1.0' encoding = 'UTF-8'?>
<!--Generated by the Oracle9i JDeveloper Web Services WSDL Generator-->
<!--Date Created: Wed Feb 25 14:12:54 CET 2004-->
```

```

<definitions
  name="pruebas.calc"
  targetNamespace="http://pruebas/calc.wsdl"
  xmlns="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:tns="http://pruebas/calc.wsdl"
  xmlns:ns1="http://pruebas/Icalc.xsd">
  <types>
    <schema
      targetNamespace="http://pruebas/Icalc.xsd"
      xmlns="http://www.w3.org/2001/XMLSchema"
      xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
    </types>
    <message name="suma0Request">
      <part name="a" type="xsd:int"/>
      <part name="b" type="xsd:int"/>
    </message>
    <message name="suma0Response">
      <part name="return" type="xsd:int"/>
    </message>
    <portType name="calcPortType">
      <operation name="suma">
        <input name="suma0Request" message="tns:suma0Request"/>
        <output name="suma0Response" message="tns:suma0Response"/>
      </operation>
    </portType>
    <binding name="calcBinding" type="tns:calcPortType">
      <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="suma">
        <soap:operation soapAction="" style="rpc"/>
        <input name="suma0Request">
          <soap:body use="encoded" namespace="pruebas.calc"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        </input>
        <output name="suma0Response">
          <soap:body use="encoded" namespace="pruebas.calc"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
        </output>
      </operation>
    </binding>
    <service name="pruebas.calc">
      <port name="calcPort" binding="tns:calcBinding">
        <soap:address location="">
      </port>
    </service>
  </definitions>

```

WSDL es un fichero XML que describe el conjunto de métodos expuestos por un webservice. Esta descripción incluye el número de argumentos, y tipo de cada uno de los parámetros de cada uno de los métodos, así como la descripción de los elementos que retornan.

Estas descripciones son las que se usan para generar los objetos proxy que usamos en los entornos de desarrollo con los que programamos webservices.

Por cada webservice, cogemos su descripción WSDL y generamos una clase con la misma interfaz (igual número de métodos y la misma signatura de los mismos) que describe el fichero. Esta clase es el proxy local del webservice.

El código local de un proxy de webservice es el encargado de construir las llamadas SOAP al servicio web y de recepcionar las llamadas SOAP de ese servicio Web.

Usando este patrón el programador es capaz de abstraerse de todos los elementos que intervienen en una llamada a un servicio Web, para él, es exactamente igual a llamar a una clase local (el proxy) y es éste el que se encarga de encapsular la complejidad propio de la comunicación con el servicio Web.

Las clases proxys son generadas por lo general de forma automatiza por la mayoría de los entornos de desarrollo.

Todo esto se puede ver en la figura 1 de este artículo.

Siguiendo con el ejemplo anteriormente mencionado, la clase Proxy que podemos generar de forma automatizada es la siguiente:

```
package pruebas;
import oracle.soap.transport.http.OracleSOAPHTTPConnection;
import org.apache.soap.encoding.soapenc.BeanSerializer;
import org.apache.soap.encoding.SOAPMappingRegistry;
import org.apache.soap.util.xml.QName;
import java.net.URL;
import org.apache.soap.Constants;
import org.apache.soap.Fault;
import org.apache.soap.SOAPException;
import org.apache.soap.rpc.Call;
import org.apache.soap.rpc.Parameter;
import org.apache.soap.rpc.Response;
import java.util.Vector;
import java.util.Properties;
/**
 * Generated by the Oracle9i JDeveloper Web Services Stub/Skeleton Generator.
 * Date Created: Wed Feb 25 14:15:09 CET 2004
 * WSDL URL: file:/C:/mywork/wsWS/calculadora/src/pruebas/calc.wsdl
 */

public class PruebasCalcStub {
    public PruebasCalcStub() {
        m_httpConnection = new OracleSOAPHTTPConnection();
        m_smr = new SOAPMappingRegistry();
    }

    public String endpoint = "";
    private OracleSOAPHTTPConnection m_httpConnection = null;
```

```
private SOAPMappingRegistry m_smr = null;

public Integer suma(Integer a, Integer b) throws Exception {
    Integer returnVal = null;

    URL endpointURL = new URL(endpoint);
    Call call = new Call();
    call.setSOAPTransport(m_httpConnection);
    call.setTargetObjectURI("pruebas.calc");
    call.setMethodName("suma");
    call.setEncodingStyleURI(Constants.NS_URI_SOAP_ENC);

    Vector params = new Vector();
    params.addElement(new Parameter("a", java.lang.Integer.class, a, null));
    params.addElement(new Parameter("b", java.lang.Integer.class, b, null));
    call.setParams(params);

    call.setSOAPMappingRegistry(m_smr);

    Response response = call.invoke(endpointURL, "");

    if (!response.generatedFault()) {
        Parameter result = response.getReturnValue();
        returnVal = (Integer)result.getValue();
    }
    else {
        Fault fault = response.getFault();
        throw new SOAPException(fault.getFaultCode(), fault.getFaultString());
    }

    return returnVal;
}

public void setMaintainSession(boolean maintainSession) {
    m_httpConnection.setMaintainSession(maintainSession);
}

public boolean getMaintainSession() {
    return m_httpConnection.getMaintainSession();
}

public void setTransportProperties(Properties props) {
    m_httpConnection.setProperties(props);
}

public Properties getTransportProperties() {
    return m_httpConnection.getProperties();
}
```

}

☞ **UDDI** es un acrónimo de Integración, Descubrimiento y Descripción Universal (en inglés Universal Description, Discovery and Integration). Es un directorio de servicios Web distribuido y basado en Web que permite que se listen, busquen y descubran este tipo de software. Podríamos compararlo con las típicas páginas amarillas.

2.- Escenarios de uso.

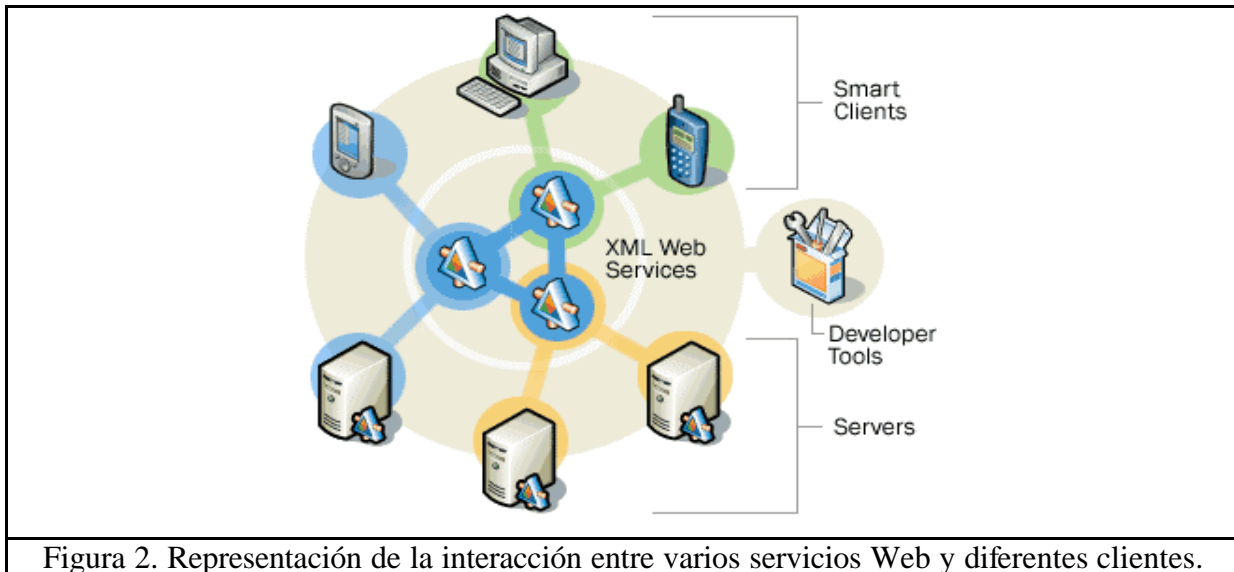


Figura 2. Representación de la interacción entre varios servicios Web y diferentes clientes.

En este apartado vamos a analizar en qué contextos son útiles los servicios Web.

Como ya comentamos anteriormente, los servicios Web no son una panacea, es decir no son la solución más adecuada para todos los problemas.

En un principio los Web Services están pensados para 3 tipos de escenarios:

- ?? Servicios Simples y públicos. Se expondría una funcionalidad simple accesible desde internet. Ejemplo: Una empresa de transporte podría exponer una función a la que se le pasasen como parámetros el lugar de origen, el lugar de destino, la hora deseada de entrega, y el peso del paquete, y obtendríamos como respuesta el precio del envío.
- ?? Integración de aplicaciones. Usamos los servicios web básicamente como extensiones de sistemas ya construidos para que éstos sean accesibles por aplicaciones y sistemas heterogéneos desarrollados bajo cualquier plataforma y lenguaje y que participen en procesos comunes. De esta forma podemos integrar software muy diverso, incluso aplicaciones que realicen transacciones de empresa-a-empresa.
- ?? Sistemas de Grid Computing. Existen problemas para los que su resolución exige tener muchos procesadores u ordenadores funcionando de forma coordinada. Lo que se hace es dividir el problema en subproblemas y resolver cada uno de estos subproblemas en un nodo de computación, finalmente con el resultado devuelto por todos estos nodos se obtiene la solución. Los sistemas de Grid Computing también están siendo sometidos a un examen concienzudo

por parte de la tecnología de los servicios Web. Estos pueden ser muy útiles para muchas aplicaciones de este tipo.

Centrémonos un poquito más en la Integración de Aplicaciones.

2.1.- Qué es la EAI.

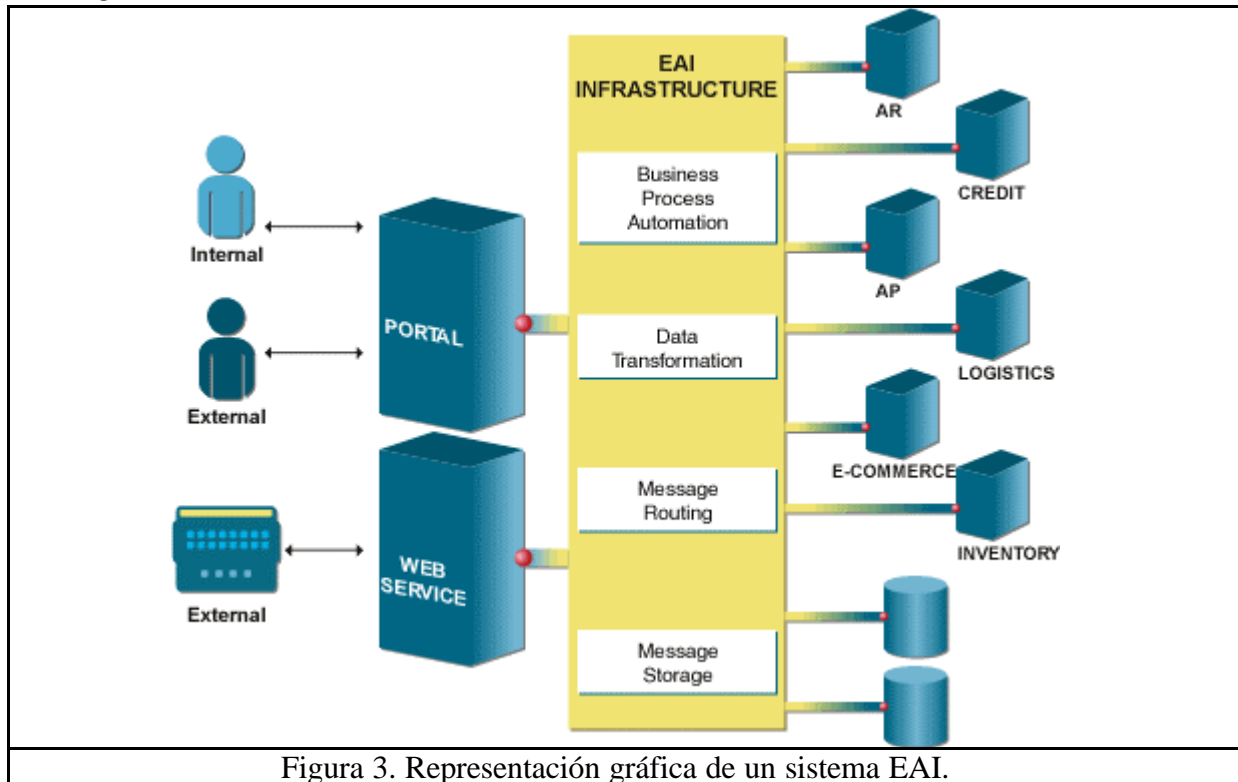


Figura 3. Representación gráfica de un sistema EAI.

El término EAI es un acrónimo de Integración de Aplicaciones Empresariales (Enterprise Application Integration). Se refiere a los desarrollos de software orientados a crear medios para compartir datos y procesos entre distintas aplicaciones de una organización.

Originalmente los programas de las empresas solían ser independientes, de tal forma que se tenía un programa para recursos humanos, otra para llevar el inventario, la contabilidad, las ventas, etc, sin interactuar entre sí. Por lo general eran desarrollados para resolver necesidades específicas, con la tecnología que en cada momento existiese y haciendo uso, casi siempre, de sistemas y tecnologías propietarias. Conforme las empresas crecieron, reconocieron la necesidad de que sus diferentes sistemas pudiesen compartir información e interactuar. Esto es lo que hacen los sistemas de EAI.

2.2.- Plataformas EAI y WebServices.

Los servidores de integración tales como SeeBeyond, Tibco, Vitria, y webMethods ofrecen soluciones que incluyen adaptadores empaquetados para interactuar con sistemas específicos (podríamos considerarlo bibliotecas de 'drivers'), junto que herramientas que simplifican el proceso de construir adaptadores específicos para otros sistemas que tenga la empresa.

El EAI lo que hace es interconectar a todos esos sistemas, de tal forma que cualquier aplicación conectada al EAI (es decir que tenga un adaptador) puede interactuar con

cualquier otra aplicación conectada al EAI.

Podríamos ver al EAI como un traductor que permite que dos sistemas que hablan en idiomas distintos sean capaces de entenderse.

Conforme ha ido pasando el tiempo, los constructores de sistemas EAI han ido añadiendo funciones avanzadas como: el mapeo de datos, la transformación y traducción de datos, la coordinación de transacciones, la gestión de comunicaciones y la gestión de procesos de negocio. Estas capacidades son críticas para las empresas de cierto tamaño en sus proyectos de integración de sistemas.

Estos productos tienen también su parte menos brillante como el coste, la complejidad, y el uso de arquitecturas propietarias.

Desde la perspectiva de la integración de aplicaciones los servicios Web (de los cuales ya hemos dicho que este es uno de sus escenarios típicos) ofrecen ventajas sobre los sistemas EAI en términos de uso de estándares, simplicidad, y bajo coste, ya que ofrecen una solución rápida y ajustada en coste para resolver problemas de interoperatividad e integración, usando infraestructuras ya existentes y reutilizando la tecnología de componentes.

Eso sí, tenemos que también ser conscientes que actualmente los servicios Web no son capaces de ofrecer todo el conjunto de características avanzadas de muchos sistemas EAI, como por ejemplo, gestión de la capa de comunicación, coordinación de transacciones, seguridad, etc, aunque son cuestiones sobre las que se está trabajando.

Los servicios Web por tanto son atractivos para soluciones de este tipo en el ámbito de la integración de baja y media complejidad.

2.3.- Qué es SOA (Service Oriented Architecture).

Últimamente cuando se habla de servicios Web se emplea también el término SOA, ¿qué es? SOA es un acrónimo de Arquitectura Orientada al Servicio (Service-Oriented Architecture, en inglés). SOA es un método para diseñar y construir soluciones software muy independientes (poco acopladas). La funcionalidad sería accesible programáticamente por otras aplicaciones a través de interfaces publicados y que puedan ser descubiertos. Los servicios Web representan una implementación de una Arquitectura Orientada al Servicio.

Básicamente una arquitectura orientada al servicio es una colección de servicios. Estos servicios se comunican entre ellos. La comunicación puede involucrar simplemente el paso de datos o la coordinación de alguna actividad entre varios servicios.

Las arquitecturas orientadas a servicios no son una cosa nueva. Para mucha gente la primera de estas surgió en el pasado con el uso de DCOM o los ORBs (Object Request Brokers) de CORBA.

Y ¿qué es un servicio? Si una arquitectura orientada al servicio debe ser efectiva, necesitamos un entendimiento claro del término servicio.

Un servicio es una función que está bien definida, es auto contenida, y no depende del contexto o el estado de otros servicios.

La tecnología de los servicios Web es la tecnología de conexión más apropiada para las arquitecturas orientadas a servicios.