XUNTA DE GALICIA

ESCOLA GALEGA DE ADMINISTRACIÓN PÚBLICA

30. ARQUITECTURA SOA. SERVICIOS WEB. TECNOLOGÍAS XML.

Fecha de actualización 26 de julio de 2011

TEMA 30. ARQUITECTURA SOA. SERVICIOS WEB. TECNOLOGÍAS XML.

30.1. INTRODUCCIÓN Y CONCEPTOS

Los sistemas actuales tienen una gran complejidad debido a la integración de múltiples componentes heterogéneos. La comunicación y relación entre estos componentes es uno de los grandes problemas actuales siendo **SOA** (en inglés *Service Oriented Architecture*) uno de los actuales modelos de solución. Esta arquitectura entiende la comunicación entre aplicaciones y componentes como servicios, no necesariamente servicios web, demandados por clientes o subscritores y proporcionados y publicados por proveedores. Las arquitecturas para servidores de aplicaciones de uso más extendido como .NET y JEE acostumbran a definir una **capa de integración** que agrupa los componentes encargados del acceso a datos, sistemas *legacy*, motores de reglas de workflow, acceso a LDAP, etc. Para la comunicación de los componentes de esta capa existen varias soluciones como JCA, JMS y servicios web, está última una de las más aceptadas actualmente.

Los **Servicios web** permiten la comunicación entre sistemas heterogéneos a través del acceso a la URL de aplicaciones empleando protocolos basados en XML como SOAP o SAAJ. Los clientes acceden al servicio a partir de su interfaz definida en presencia de WSDL (en inglés *Web Service Definition*

Language) o insertada en algún registro de servicios web.

El uso de XML se convierte en un estándar de integración, siendo la base de las comunicaciones en esta capa, tanto para estructurar como para almacenar e intercambiar información, extendiendo su uso a otros ámbitos. Las **tecnologías XML** son un conjunto de módulos que ofrecen servicios como: XSL/XSLT para diseño de documentos, Xpath como lenguaje de rutas para acceso a documentos, el lenguaje de consulta XQL, y otros como XLink o XPointer.

30.2 ARQUITECTURA SOA

SOA define una arquitectura orientada a servicios que busca simplificar el modelo de integración de sistemas distribuidos heterogéneos. En esta arquitectura los componentes publican e invocan servicios en la red a través de mecanismos de comunicación como JCA, JMS, SOAP, RPC o Servicios web. Los servicios son funcionalidades de la lógica de negocio que pueden invocarse de manera remota para obtener un resultado. Se definen en presencia de una interfaz explícita, como por ejemplo a través de WSDL, independiente de su implementación empleando estándares de comunicación basados en XML. SOA define tres **bases** fundamentales:

1) **Orientación al intercambio de mensajes.** La base del sistema es la comunicación entre los nodos del sistema.
2) **Abstracción de componentes.** Cada sistema se reduce a su interfaz y el conjunto de servicios que define, con lo cual permite la integración entre cualquier tipo de sistema.
3) **Metadatos.** Descripciones e información asociada a servicios y mensajes, mejorando la capacidad semántica del sistema.

A nivel lógico los principales componentes en una arquitectura SOA son:

a) **Servicios**. Entidades o funcionalidades lógicas definidos en interfaces públicas, que pueden o no requerir autenticación.

b) **Proveedor de servicios**. Componente software que implementa un servicio y publica su interfaz.
c) **Cliente de servicios**. Componente software que invoca un servicio de un proveedor.
d) **Localizador de servicios**. Proveedor de servicios que registra las interfaces y permite a los clientes buscar en el registro y acceder a su localización.
e) **Servicio de interconexión**. Proveedor que comunica solicitudes de servicio a otros proveedores.

El concepto de BPM o Gestión de procesos de negocio (en inglés *Business Process Management*), está muy relacionado con SOA. BPM es un modelo de gestión centrado en procesos de negocio y de cómo integrar sus funcionalidades en sistemas heterogéneos. A partir de la identificación y gestión de los procesos de la organización puede implantarse una solución BPM a través de una arquitectura SOA. Fruto de esta idea aparecen soluciones como:

- ✓ **BPMN**. Nota para el modelado de procesos de negocio.
- ✓ **BPEL**. Lenguaje de ejecución de procesos de negocio con servicios web para la orquestación de servicios. Generalmente se realiza una conversión de BPMN a BPEL.
- ✓ **BPEL4WS.** Lenguaje de definición y ejecución de procesos de negocios empleando servicios web (en inglés *Business Process Execution Language for Web Services*). BPEL4WS es resultado de la convergencia de WSFL (en inglés *Web Services Flow Language*) y XLANG, permitiendo componer Servicios web como servicios compuestos denominados Servicios de negocio.

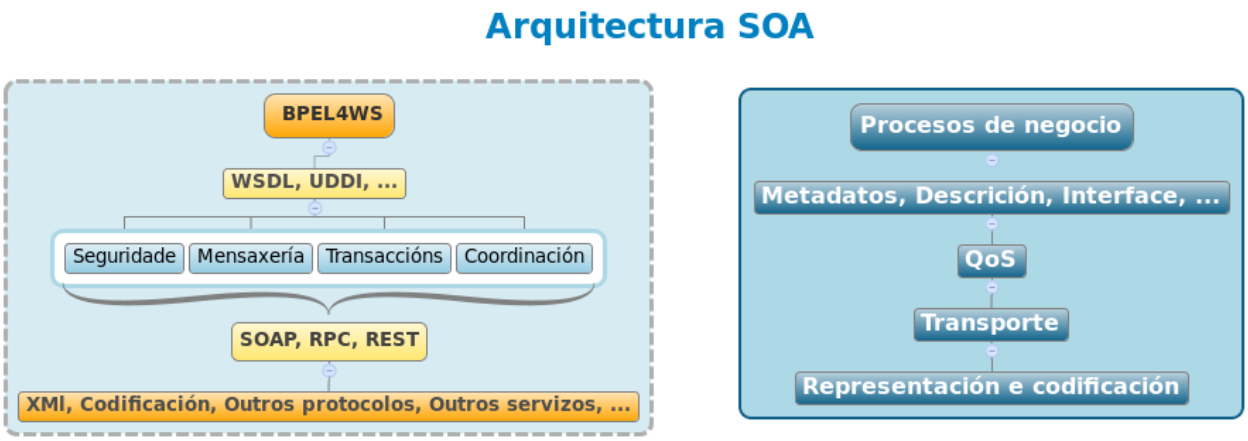
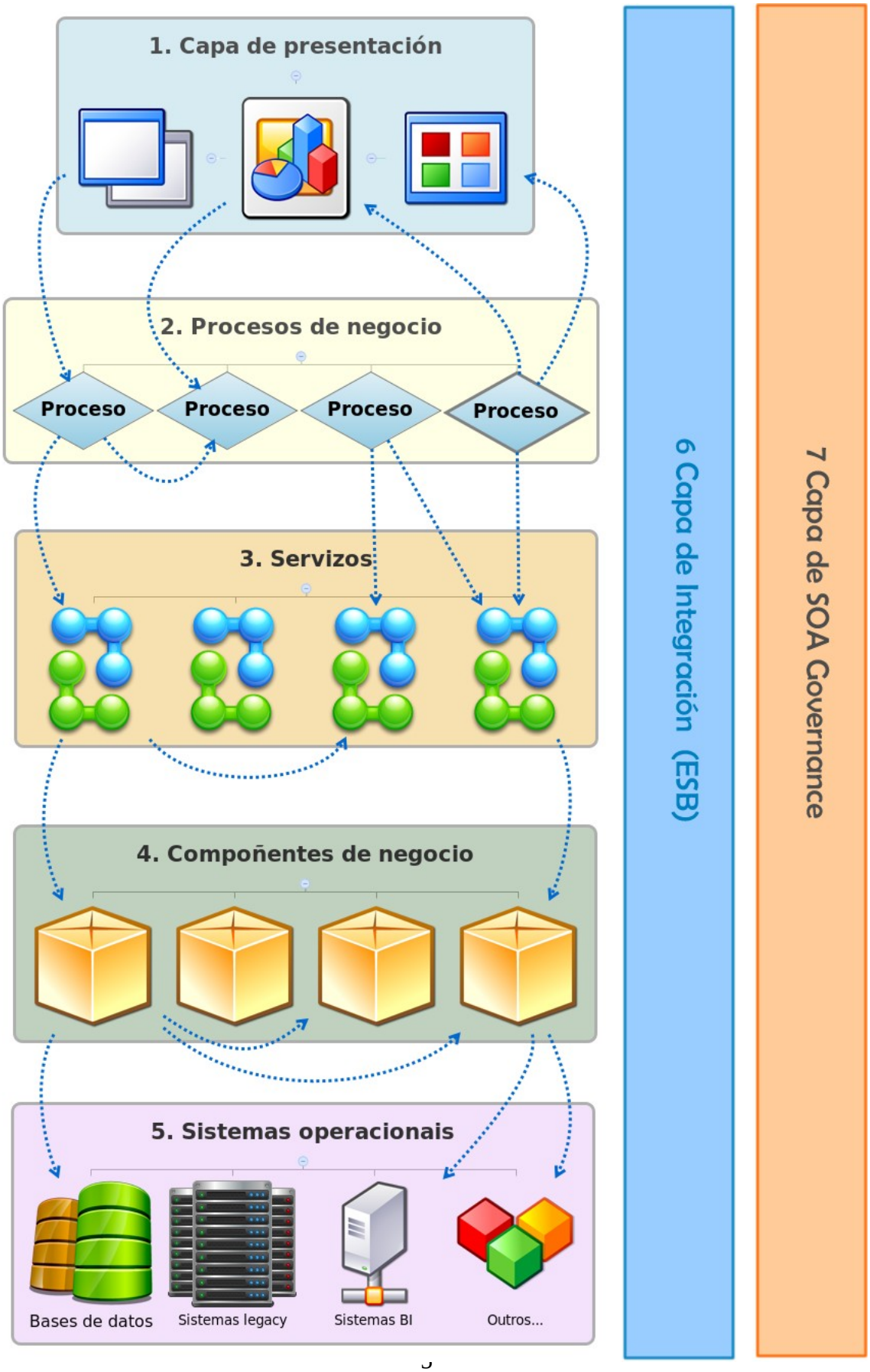


Figura 1: Arquitectura SOA.

La integración en la arquitectura SOA de los BPM permite establecer lo que se da en llamar **SOA Governance,** una estructura para la toma de decisiones y establecimiento de responsabilidades en la organización a través de la implantación de políticas, monitorización de servicios, incorporación de buenas prácticas, principios arquitectónicos, avance continuo de los procesos de negocio, etc. en definitiva análisis y diseño de soluciones que permitan cumplir con éxito la implantación de SOA en una organización.

En el aspecto **conceptual SOA** describe una serie de guías o patrones para servicios alineados con un modelo de negocio. Un modelo conceptual de SOA permite definir un diseño en múltiples capas donde los servicios se relacionan con procesos de negocio y sistemas de información. La división más habitual considera 7 capas diferenciadas, que se pueden ver gráficamente en la Figura 2:

1) **Capa de Presentación**. Interfaces de usuario en sitios web, aplicaciones y portales que invocan funcionalidades de los procesos de negocio.


1. Capa de presentación
2. Procesos de negocio
Proceso
Proceso
Proceso
Proceso
3. Servizos
4. Compoñentes de negocio
5. Sistemas operacionais
Bases de datos
Sistemas legacy
Sistemas BI
Outros...
6 Capa de Integración (ESB)
7 Capa de SOA Governance

2) **Capa de Procesos de negocio.** Representa los procesos y flujos operativos o workflows que invocan los clientes desde la capa de presentación u orquestación de servicios. Los procesos por norma general se representarán con BPEL y se implementarán con alguna herramienta de transformación.
3) **Capa de Servicios.** Funcionalidades de los componentes de la lógica de negocio que se publican para uso de los clientes. Se referencian a partir de la interfaz siendo transparentes a su implementación.
4) **Capa de Componentes de negocio.** Son los encargados de proporcionar las funcionalidades que se publicarán en los servicios así como cualquier otra intermedia o común al sistema. A este nivel irían los servidores de aplicaciones, otros servicios web, aplicaciones, paquetes y librerías.
5) **Capa de Sistemas operacionales.** En este nivel irían los sistemas de información de la organización, sistemas *legacy*, sistemas CRM o ERP, aplicaciones de BI, etc...
6) **Capa de Integración.** Agiliza la integración de servicios a través de sistemas tipo Buses de Servicios Empresariales o ESB, que realizan funciones de enrutamiento, monitorización y administración, transformaciones de los mensajes, etc... dentro del área de comunicaciones.
7) **Capa de SOA Governance.** Realiza funciones de administración, monitorización y control de la calidad del servicio en áreas como seguridad, disponibilidad y otros factores generales no recogidos en la capa de integración.

Existen **patrones de diseño** tomando como punto de partida esta arquitectura, se trata de patrones para Servicios web y patrones **POSA** (en inglés *Service-Oriented Architecture Patterns*). Algunos de los principales son:

- **Service Oriented Architecture.** Patrón que define la arquitectura SOA estableciendo reglas, relaciones y dependencias entre los componentes del sistema. Permite buscar servicios dinámicamente con independencia de la plataforma y sin requerir implementación, con transparencia. Este patrón es una variante ampliada del **Broker Pattern** de POSA. En este patrón un Servicio o nodo intermedio ayuda a localizar el servicio y puede obligar a realizar todas las comunicaciones a través de él o bien una vez establecida dejar que esta sea directa entre el cliente y el servicio.
- **Architecture Adapter.** Patrón genérico que facilita la comunicación entre diferentes arquitecturas gracias a la independencia de usar XML/SOAP y la generación de clases proxy. Este patrón es implementado por *frameworks* para Servicios web como Apache Axis (Java).
- **Service Directory.** Facilita la localización de Servicios web a partir de una especificación fuerte de las interfaces a través del catálogo UDDI de interfaces WSDL.
- **Service Factory.** Permite la selección de servicios del proveedor aislando el código de comunicación UDDI. Del mismo modo el patrón de extendido Service Factory Cache hace funciones de caché en el servicio. Simplifica en parte a API del patrón Service Directory.
- **Service Facade.** Proporciona un servicio web controlador que actúe como punto de entrada de la lógica de negocio u objeto de fachada. Puede emplear simultáneamente otros mecanismos de comunicación como CORBA.
- **Event Monitor.** Se emplea para notificar que un Servicio web de larga duración invocado remotamente completa la solicitud. Cuando el Servicio no dispone de mecanismos de notificación hace falta establecer un intermediario.
- **Business Object.** Un BO engloba un concepto del dominio, equiparable a un VO para entornos distribuidos.

- **Business Process.** Un BP engloba un proceso de la lógica de negocio, representando la jerarquía formada por las diferentes implementaciones de sus funcionalidades y la interfaz del servicio.
- **Asynchronous Business Process.** Este patrón se encarga de gestionar la llamada y notificación de respuesta al cliente cuando estas pueden ser de larga duración.
- **Business Object Collection.** Agrupa diferentes procesos de negocio en un mismo BOC.
- **Observer Services.** Se basa en un registro de servicios donde el observador notifica al cliente sobre eventos que derivados de los servicios en los que esta registrado.
- **Publish/Suscribir Services.** Evolución del patrón Observer Services incorporando un sistema de notificaciones para sustituir al registro. Se emplea cuando los servicios web no incorporan un sistema de notificación y precisan un intermediario.
- **Data Transfer Object.** Permite enviar múltiples objetos en una misma llamada reduciendo el número de conexiones.
- **Partial Population.** Permite que los clientes seleccionen parte de la información del mensaje de respuesta a la solicitud de servicio buscando un mejor aprovechamiento del ancho de banda.
- **Microkernel**. Separa un núcleo de funcionalidad mínimo de partes especificadas por el cliente.
- **Web Service Interface.** Proporciona una interfaz que puede emplearse desde los clientes para invocar los métodos de un proxy de Servicio web genérico en lugar de depender de la clase proxy generada a partir de la WDSL.

REST (en inglés *Representation State Transfer*) representa un modelo de comunicación donde cada petición HTTP contiene la información necesaria para responder a la petición sin tener que almacenar el estado de la sesión. En REST todos los servicios son recursos, identificados por URIs y se

diseñan sus representaciones mediante XML, JSON o microformatos. REST representa una arquitectura SOA que no hace uso de Servicios web, SOAP ni RPC.

Actualmente dentro del marco de la Web 2.0 surge una nueva variante en las arquitecturas SOA, el concepto de **Mashup** , un sitio o aplicación web que hace uso de contenido de otras aplicaciones o servicios vía HTTP. Este contenido es recuperado en un modelo de Servicios web a través de su API pública evitando caer en el Web Scraping. Para emplear los Mashups como XML se emplean lenguajes específicos como EMML (en inglés *Enterprise Mashups Markup Language*). Las arquitecturas Mashup constan de tres componentes:

- ✓ **Los proveedores de servicios.** Orígenes de datos que publican a través de una interfaz los métodos de acceso a los mismos y permiten su consulta vía Atom, RSS, REST, JSON, Bases de datos o interfaces WSDL de Servicios web.
- ✓ **Aplicación o Servicio web Mashup.** Proporciona un nuevo servicio a partir de la información obtenida de los proveedores.
- ✓ **Clientes.** Usuarios finales, u otras aplicaciones o servicios que hacen peticiones al Mashup. En los clientes acostumbran a emplearse tecnologías RIA del tipo de AJAX o Comet.

Otro concepto que se puede relacionar con SOA es el de la **Nube** (en inglés *Cloud Computing*). La nube se fundamenta en emplear la red Internet para publicar servicios, que pueden o no requerir identificación. En la nube todo son servicios, aplicaciones, bases de datos, redes, y se gestionan y se acceden como tal. La arquitectura de la nube se estructura habitualmente en tres capas:

1) **Software como servicio** o SaaS (en inglés *Software as a Service*). Sería el nivel más alto, orientado a los usuarios y clientes finales. Se incluirían las aplicaciones propias y aplicaciones de terceros del estilo

de Google Aps. La misma infraestructura del proveedor sirve a múltiples organizaciones finales.

2) **Plataforma como servicio** o PaaS (en inglés *Platform as a Service*). Serían la capa intermedia encargada de encapsular sistemas o *middleware* permitiendo correr aplicaciones sobre ellas. Ejemplos de este servicio serían Google App Engine o Windows Azure. De este modo una organización externa proporciona un servicio de infraestructura y soporte para otras organizaciones.
3) **Infraestructura como servicio** o IaaS (en inglés *Infrastructure as a Service*). En este caso se ofrecen servidores para computación, red, almacenamiento o bases de datos a través de diferentes técnicas como por ejemplo a través de máquinas virtuales.

El **Bus de Servicios Empresariales** o ESB (en inglés *Enterprise Service Bus*) representa otras de las características de SOA, aunque no es imprescindible en una arquitectura de este tipo. Se trata de un componente que hace abstracción del sistema de mensajes de la organización a través de un sistema único para todos los elementos de un sistema SOA. El ESB proporciona funciones de transformación, adaptación, conexión y enrutamiento que pueden ser implementadas en SOA. En una arquitectura SOA con un ESB todas las aplicaciones y servicios se conectan a un punto único y central que administra las comunicaciones, realizando funciones de *middleware*. El ESB se construye sobre tecnologías XML, XSLT, XPath, JMS o propias de Servicios web. Hace uso de elementos denominados Contenedores de Servicios o Brokers que hacen la función de servidores de comunicaciones. Entre los servicios que proporciona se encuentran:

- ✓ Funcionalidades de enrutamiento, fraccionamiento y combinación de mensajes bajo la base de los patrones EIP (en inglés *Enterprise Integration Pattern*).
- ✓ Funciones de supervisión y control de la calidad del servicio a través de Acuerdo de Nivel de Servicios o SLA de los servicios.

- ✓ Funciones de monitorización, seguridad y mediación de protocolos.

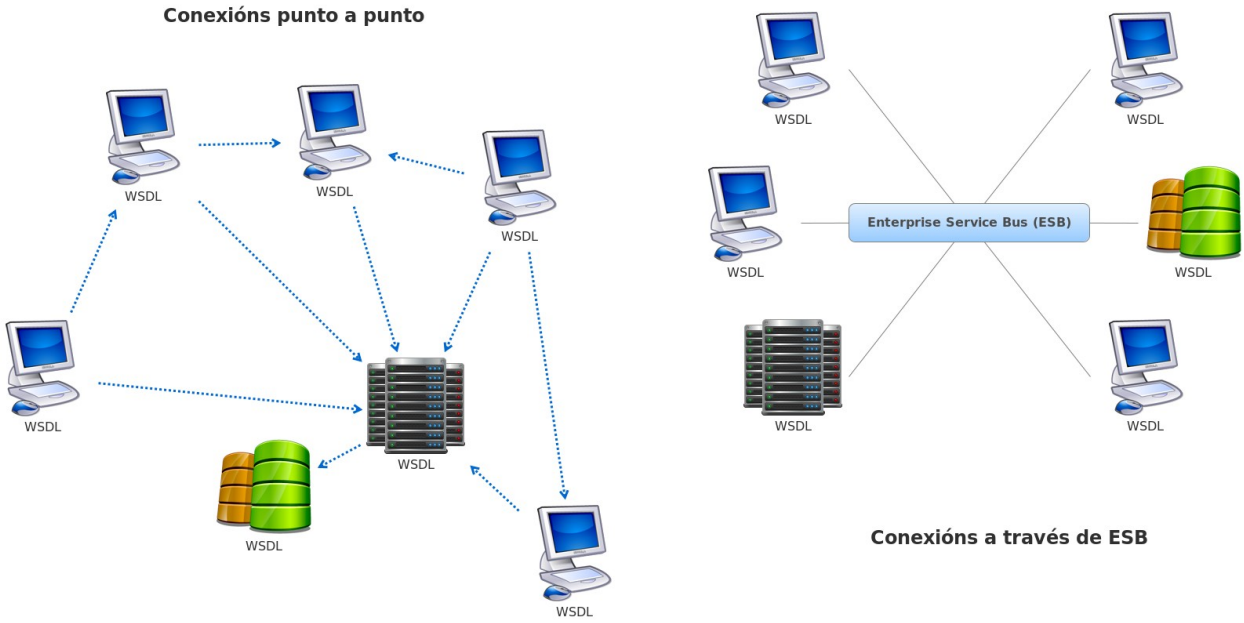


Figura 3: Bus de Servicios Empresariales (ESB).

El Bus sustituye la comunicación directa entre dos aplicaciones o servicios, de modo que la comunicación se hace de manera transparente a través del ESB. Emplea un sistema de mensajería, como por ejemplo Tibco, soportando varios MEP o patrones de intercambio de mensajes, así como colas para reenviar las peticiones a los proveedores de servicios y las respuestas a las solicitudes a los clientes. Existen *frameworks* que recogen los componentes precisos para implantar un ESB como Mule ESB, este es un *framework* ligero destinado a mensajería y control de eventos. Permite integraciones con otros *frameworks* como Struts o Spring y soporta muchos componentes de servicio como JMS, SOAP, BPEL, JBI (en inglés Java Business Integration), y otros.

Por último señalar que las arquitecturas SOA pueden completarse con módulos específicos según las necesidades de la organización, como:

- ✓ **Seguridad**. Con la adopción de diferentes tecnologías, SSL, Kerberos, X.509, Firmas XML, Encriptación XML, XML Canonicalization, SAML (en inglés *Security Assertion Markup Language*) o XKMS (en inglés *XML Key Direction Specification),* que administra la llave pública o PKI de las infraestructuras.
- ✓ **Orquestación y coreografía de servicios.** En este modelo la interacción entre servicios no se produce directamente sino que se define una entidad que define la lógica de interacción, facilitando la colaboración que será un servicio de control primario. En BPEL el servicio primario será un proceso BPEL, pero también se puede definir con BPEL4WS, WSFL o XLANG. Mientras la orquestación precisa de un director de orquesta o servicio central, el modelo de coreografía establece interacciones punto a punto a partir de reglas de colaboración generales. Para la coreografía existen lenguajes específicos como WS-CDL (en inglés *Web Services Choreography Description Language*) que tienen definida la forma de representar las interacciones.
- ✓ **Gestión transaccional.** Existen varias tecnologías que coordinan las transacciones entres servicios autónomos. El BTP (en inglés Business Transaction Protocol) donde ninguno de los servicios gestiona una transacción, sino que esta se comunica a todos y deciden si se unen o no, con comunicaciones basadas en XML en un formato propio. Otros mecanismos como WS-Transaction y WS-Coordination se encargan de gestionar transacciones llevadas a cabo por varios servicios a la vez, con protocolos SOAP y WSDL. Por su parte JEE disponen de la especificación JAXTX para transacciones complejas con el objetivo de aislar estas de los contenedores.

30.3 SERVICIOS WEB

Los Servicios web son uno de los modelos de implementación de SOA. Un

Servicio web que proporciona un servicio vía web en una red a través de una interfaz que le permite recibir peticiones y transmitir respuestas. Para soportar este sistema se desarrollaron una gran variedad de protocolos y tecnologías. Los principales son el HTTP/HTTPS para peticiones y respuestas y el XML como formato de intercambio. Los principales **componentes** comunes a los servicios web serían:

a) **SOAP** (en inglés *Simple Object Access Protocol*). El protocolo de comunicación, sobre la capa de transporte basado en XML, que sirve para invocar los servicios a través de un protocolo siendo los más habituales HTTP o SMTP, pero realmente es independiente y permite otros cómo POP3 o JMS. Permite tanto describir el contenido del mensaje y reglas de codificación de los tipos de datos, como aspectos de seguridad y transaccionalidad. Se encuentra estandarizado por el W3C, lo que garantiza la comunicación entre sistemas heterogéneos que lo implementen.
b) **UDDI** (en inglés *Universal Description, Discovery and Integration*). Directorio donde se publican los servicios proporcionando la información necesaria para permitir su invocación. Presenta dos API que permiten a los servicios publicar sus funcionalidades y a los clientes enviar las peticiones y obtener los resultados. Cada servicio se publica en el UDDI proporcionando la URL de su WSDL y meta-información. De manera general se entiende que UDDI proporciona tres tipos de servicios: información general sobre los proveedores de los servicios (páginas blancas), categorías y clasificaciones de servicios (páginas amarillas) y las reglas de negocio o información técnica sobre los servicios (páginas verdes).
c) **WSDL** (en inglés *Web Services Description Language*). Lenguaje basado en XML y XML Schema que permiten la descripción de la interfaz de los Servicios web y que está estandarizado por el W3C. En un documento WSDL se definen los tipos de datos, los mensajes, los

endpoints, los *bindings* y los servicios.

d) **Serialización de datos.** Se emplean definiciones de XML Schema para especificar cómo codificar los datos en conjunción con las reglas de codificación de SOAP. Aunque el mecanismo más habitual sea el SOAP Document/Literal existen otros mecanismos como: RPC/Encoding, Document/Encoding o RPC/Literal.

Según lo visto anteriormente para las arquitecturas SOA en general, se pueden definir varios tipos de servicios según su complejidad, comenzando por los de nivel básico a los de niveles más complejos.

	Servicios de nivel básico	**Servicios de alta complejidad**
Función	Integración de la funcionalidad de una aplicación	Elemento llave de una arquitectura SOA
Protocolos y tecnologías	SOPA, UDDI, WSDL	ebXML, BPEL, BTP, RossetaNet, Apache Axis, ...
Tipo de contenido	Plano	MIME, PDF, ...
Comunicaciones	Punto a punto	Multiparty, ESB, ...
Mensajería	JMS, RPC, ...	Colaboración y *workflows*
Transaccionalidad	No transaccional	Transaccional
Seguridad	SSL, autenticación, ...	Firma digital, XML-encryption, Kerberos, ...

Tabla 1: Complejidad de los servicios web.

Según el tipo de comunicaciones las APIs más habituales son:

a) **API de mensajería.** Clientes y servicios disponen de sistemas de mensajería que les permiten comunicarse en formato XML. Al estar

orientadas hacia los sistemas de mensajería presentan una alta QoS.

b) **API de RPC.** La solución más habitual, que emplea un compilador intermedio de WSDL para generar el *stub* y el *skeleton* para cliente y servidor respectivamente, tal y como sucede con CORBA. Este sistema es lo que acostumbran a emplear los *frameworks* actuales como Apache Axis.

c) **API para servidores de aplicaciones (JEE/.NET).** Estas API vienen disponibles en las bibliotecas de clases de cada arquitectura, como por ejemplo en JEE se dispone de: JAXM (en inglés *Java API for XML Messaging*) para intercambio de mensajes; JAX-RPC (en inglés *Java API for XML-based RPC*), que permite enviar peticiones remotas a terceros y recibir resultados; y JAXR (en inglés *Java API for XML Registries*), que proporciona acceso a registros de negocio y mecanismos para compartir información.

El **proceso de implementación de un servicio web** consiste en implementar las funcionalidades del servicio reutilizando las clases generadas a partir de un WSDL o de una API (JAX-RPC, Axis,...). Se pueden aprovechar las herramientas existentes en los ID, u otras más específicas con Ant o *Wsdl2java*. Una vez implementadas las clases con la lógica del servicio se generan las clases en un war y se despliegan en un contenedor de Servlets o en un IIS. Sobre el modelo de programación se emplean diferentes variantes:

a) **Estilo CORBA.** Se generan todas las clases al compilar empleando clases de las API (Axis, JAX-RPC, ...)

b) **Dynamic Proxy.** La interfaz WSDL se crea al compilar, pero el proxy en el cliente sólo se compila en tiempo de ejecución.

c) **Dynamic Invocation Interfaz.** Tanto WSDL como cliente se generan en tiempo de ejecución. El cliente busca e invoca el servicio vía

broker.

Otro de los factores a considerar es el tema de la **seguridad** en los Servicios web, que por la propia naturaleza de las arquitecturas SOA resulta un tema complejo. Los principales elementos de seguridad en lo que respecta la JEE, aunque muchas serían extensibles la .NET, serían:

- ✓ Para JAX-RPC a API **XWS-Security** que facilita la integración de aspectos de seguridad.
- ✓ El estándar **XML-DigitalSignature** para firma digital.
- ✓ El estándar **XML-Encrytion** para encriptación de mensajería.
- ✓ **Certificados X.509** para autenticación.
- ✓ Bases de datos de certificados basadas en **JKS** (en inglés *Java Key Store*).

Dentro de la arquitectura los diferentes mecanismos se integrarían nivel a nivel de la siguiente manera:

1) **Nivel de transporte.** Autenticación básica, autenticación por certificado vía SSL/TLS. Codificación de usuario/contraseña en los *stubs* y reglas de seguridad para *endpoints.*
2) **Nivel de mensaje.** Firma de contenidos con certificados XML-DigitalSignatura, certificados X.509 y encriptación**.**

Entre las **posibilidades** existentes para implementar Servicios web se encuentran:

- ✓ APIs Java: JAX-RPC, JAXM, SAAJ (mensajes SOAP como objetos), JWSDL (Acceso a descripciones WSDL), JAXR (Acceso al UDDI), *framework* Apache Axis, ...
- ✓ .NET: ASP .NET, MS SOAP Toolkit,...
- ✓ Otras tecnologías: NuSOAP para PHP, Axis para C++,...

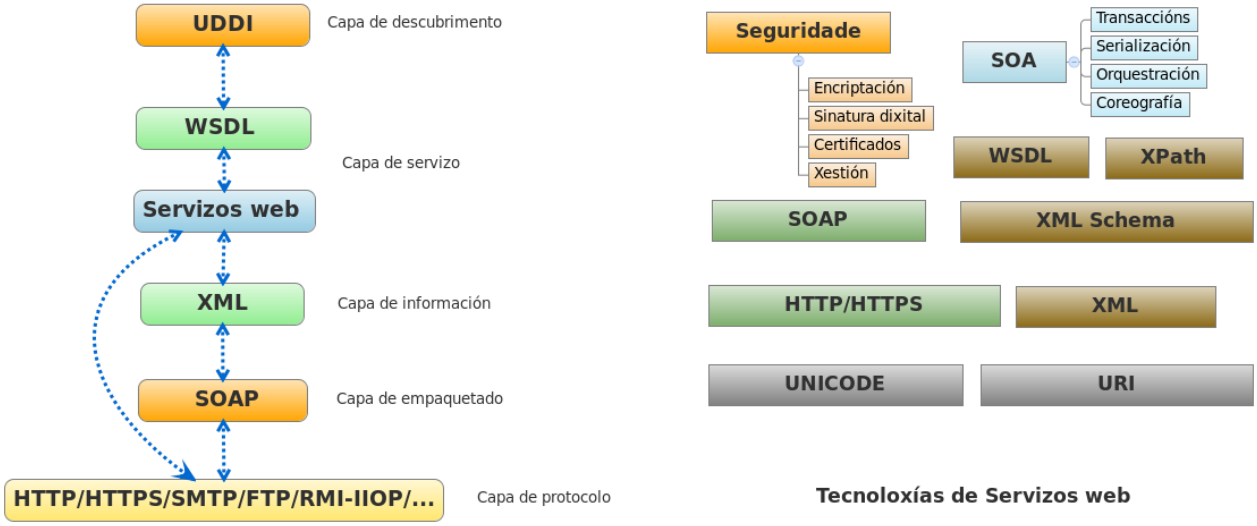


Figura 4: Tecnologías de Servicios web.

30.4 TECNOLOGÍAS XML

El lenguaje XML (en inglés *xtensible Markup Language*) es un metalenguaje para etiquetado desarrollado por el W3C. En SOA el XML representa el estándar para intercambio de información estructurada entre sistemas heterogéneos. En esencia, XML es un lenguaje de marcas que permite la creación de otros lenguajes de marcas, con diferentes usos en SOA, cada uno de estos lenguajes se denomina Aplicación XML y representa un modelo de datos de acuerdo a un esquema semántico.

El XML resulta más estricto que otros lenguajes como HTML, admitiendo varios mecanismos de validación o corrección:

- ✓ **Formación**. Un documento XML se denomina “bien formado” cuando sigue las reglas léxicas (tipo de caracteres, codificación,...) y sintácticas (anidación correcta, marcas de apertura y cierre de estructura,...) debidas.
- ✓ **Validación**. Un documento XML se considera “validado” cuando

cumple un conjunto de reglas y limitaciones denominadas en conjunto gramática o Esquemas XML (en inglés XML *Schema*). Existen varios formatos para gramáticas: los DTD heredados del SGML, los XML-Schema, que son recomendación del estándar por el W3C y otros más específicos como el XML Data.

Las **tecnologías principales más empleadas** del modelo XML en arquitecturas SOA vienen dadas por:

- ✓ **XML Schema.** Lenguaje de esquema para describir la estructura y reglas de validación de un documento XML. Se diferencia del DTD en que permite un gran número de tipos de datos. Los documentos de esquema son de extensión XSD (en inglés XML Schema Definition). La programación de esquemas se basa en los espacios de nombres y los elementos y atributos que contienen. Después de validar un documento contra un XSD se puede expresar su estructura y contenido en términos del modelo de datos del esquema. Esta funcionalidad se denomina PSVI (en inglés *Post Schema Validation Infoset*), y permite transformar el documento en una jerarquía orientada a objetos.
- ✓ **XSL**. XSL funciona como un lenguaje avanzado para crear hojas CSS transformando y realizando otras operaciones sobre documentos XML, dándoles formato. A su vez puede descomponerse en tres lenguajes o dialectos XML, todos consejos del W3C, que integran la familia XSL:
  - ✓ **XSLT** (en inglés *Extensible Stylesheet Language Transformations*). Estándar para documentos XML que permiten transformar documentos XML en función de modelos de una sintaxis a otra permitiendo estructuras de programación, funcionando a modo de intérprete. Las reglas de los modelos se definen programáticamente y la principal capacidad de este lenguaje es que permite separar el contenido de la presentación, o diferentes

presentaciones en documentos XML lo que se adapta perfectamente a los modelos de separación en capas vistos.

- ✓ **XSL-FO** (en inglés *Extensible Stylesheet Language Formatting Objects*). Documentos XML que especifican formatos de datos u objetos para su presentación. La utilidad básica de estos documentos es la presentación, con lo cual se complementan con XSLT en la salida de datos de las aplicaciones. Permite la generación de documentos multiformato: XML, (X)HTML e incluso PDF. Existen procesadores específicos para este tipo de operaciones como Apache FOP.
- ✓ **XPath o XML Path Language**. Permite identificar partes de un documento XML, accediendo a sus atributos y elementos como si fuesen nodos, a través de la construcción de expresiones que recurren y procesan un documento XML. En XSL permite seleccionar y recorrer el documento XML de entrada de la transformación, pero por extensión tiene otros muchos usos actualmente, sirviendo de base para otros lenguajes XML.

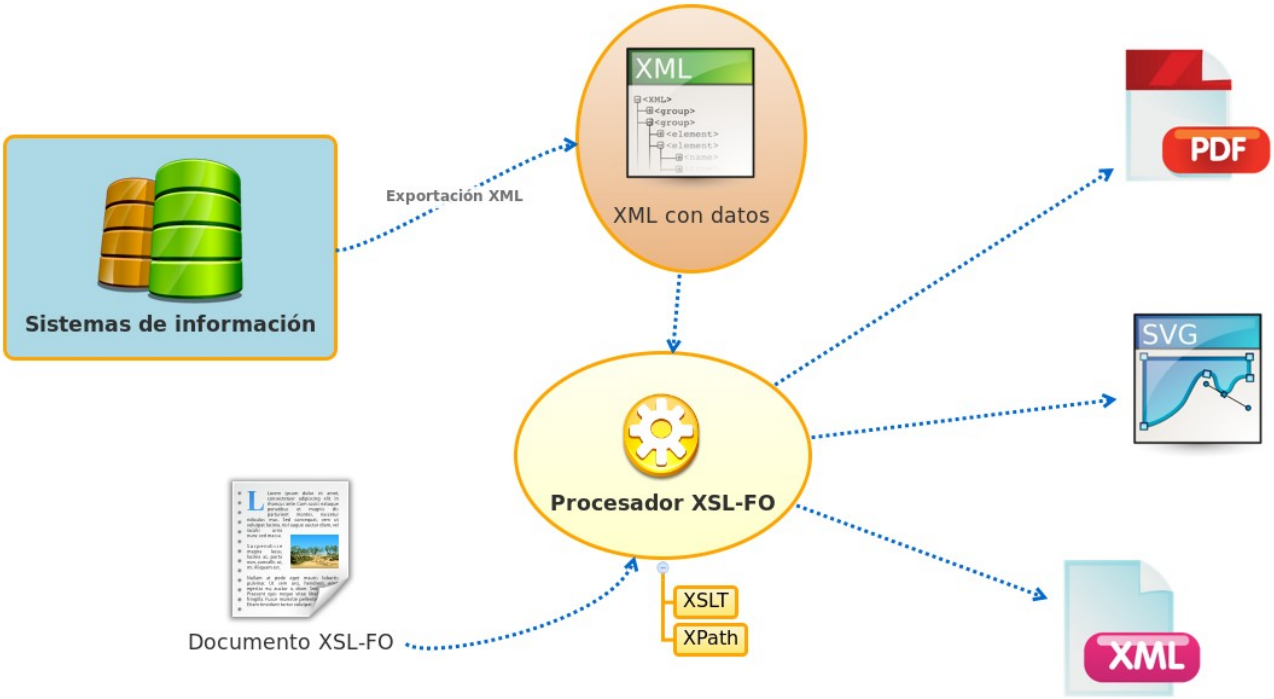


Figura 5: Familia XSL.

- ✓ **XPointer**. Recomendación del W3C que permite localizar puntos concretos o fragmentos en un documento que expande las funcionalidades de XPath a través de rangos.
- ✓ **XLink.** Recomendación del W3C que define un mecanismo para añadir hiperenlaces en archivos XML u otros recursos, con la opción de navegar en los dos sentidos, con enlaces bidireccionales o varios archivos enlazados, multienlaces. En XLink todo en la red es un recurso, y se puede enlazar desde un localizador, definiendo las relaciones entre recursos con arcos. Asimismo permite agregar a un vínculo información sobre si mismo como metadatos.
- ✓ **XQuery.** Lenguaje de consulta desarrollado por el W3C para recuperar colecciones de datos XML, con muchas similitudes con SQL. Permite extraer y manipular información de documentos XML o cualquier otro sistema de información que permita representación vía XML como Bases de datos o documentos ofimáticos. Emplea XPath

para acceder a los documentos añadiendo unas expresiones propias denominadas FLWOR. Asimismo realiza transformaciones de documentos XML y búsquedas de elementos textuales en la web o en recursos XML/(X)HTML. En las arquitecturas SOA resultan especialmente útiles para recuperar información de Bases de datos y presentarla a través de Servicios web.

- ✓ **XForms.** Lenguaje de definición de Interfaces de usuario desarrollada por el W3C, centrada especialmente en la parte de formularios web y su integración en documentos (X)HTML, ODF o SVG. Se aplica el paradigma de separar el contenido, propósito y estructura. En aplicación del MVC incorpora un modelo declarativo de compuesto de reglas y validación para datos y tipos de datos de los formularios, así como envío de parámetros; una capa de vista compuesta de los controles de la interfaz de usuario; un controlador para orquestar las manipulaciones de datos, interacciones entre el modelo y la vista y envíos de datos. Otras evoluciones de esta tecnología serían AJAXForms y XSLTForms, incorporando AJAX y XSLT a esta tecnología. Por otra parte, existen otros lenguajes relacionados con las interfaces de usuario que siguen dialectos de XML, muchas de ellas con capacidad para interactuar con XForms como: XAML, XUL, UIML, UsiXML, AUIML,...

El grupo de tecnologías anteriores podrían definirse como lenguajes XML de propósito general. En muchas ocasiones el XML se emplea de manera concreta para representación de datos complejos o con necesidades específicas para nuestro dominio. En la tabla 2 se recogen algunos ejemplos de lenguajes XML empleados para representación o adaptación de información a necesidades concretas, bien para entornos de trabajo como

XHTML y WML o bien en dominios específicos como aplicaciones de información geográfica o diseño gráfico.

	Función
XHTML	HTML con especificaciones más estrictas para presentar una mayor compatibilidad con la web semántica y los otros estándares XML
MathML	Expresar planteamientos matemáticos
SVG	Especificación para describir gráficos vectoriales y animaciones
SMIL	Permitir la integración multimedia en XHTML y SVG
WML	Adaptación del HTML para móviles y PDA
VoiceXML	Convertir habla en XML a partir de gramáticas de reconocimiento de voz
SSML	Para habla sintética
GML/KML	Para sistemas de modelado e información geográfica
X3D	Representación de gráficos en 3D
EBML	Para almacenar jerarquías de datos en formato binario de longitud variable

Tablas 2: Lenguajes XML complementarios.

El uso tan extendido del XML obliga a disponer de herramientas que permitan el tratamiento sencillo de los documentos, recorrer, manipular, procesar, etc... Muchas tecnologías disponen de *frameworks* específicos para **tratamiento de XML**:

✓ **DOM** (en inglés *Document Object Model*). Especificación del W3C de una API (org.w3c.dom) para manipular documentos XML/HTML, acceder a su contenido, estructura y estilos, a través de un analizador sintáctico. DOM genera un árbol jerárquico en memoria donde almacena todo el documento. A través de un procesador se permite acceder a cualquier nodo del árbol, o insertar/eliminar nuevos nodos.

El principal inconveniente de este modelo es que precisa gran cantidad de memoria por la necesidad de cargar todo el documento, pero tiene las ventajas de ser muy sencillo de implementar y de permitir la generación de XML. Se apoya en tecnologías XSLT y Xpath. Frameworks como Xerces se basan en DOM para tratamiento de XML así como otros basados en AJAX del tipo de Jquery, Prototype, Dojo, etc... Asimismo existen alternativas recientes similares a DOM, diseñadas explícitamente para JEE que resultan más fáciles de emplear, JDOM (org.jdom) y DOM4J (org.dom4j).

✓ **SAX** (en inglés *Simple API for XML*). API inicialmente para Java (org.xml.sax), pero que después evolucionó a otros lenguajes como C++, Perl, Phyton,..., que disponen de un analizador que genera eventos al conseguir puntos llave del documento analizado. Recorre el documento de manera secuencial a través de un administrador de eventos, el DocumentHandler, evento a evento, con lo cual no precisa cargar el documento en memoria pero no permite vuelta atrás sin ir de nuevo al inicio. Esto lo hace muy adecuado para documentos de gran tamaño. Otros *frameworks* más completos se basan a su vez en SAX, como: Xerces, Crimson, Piccolo u Oracle XML Parser.

✓ **StAX** (en inglés *Streaming API for XML*). Define un analizador sintáctico de flujo de datos integrado en JEE, con soporte para generación de XML. Se emplean dos estilos de análisis Cursor API e Iterador Event Iterator API, ambos basados en iteraciones para solventar las limitaciones de SAX y DOM. En este modelo el documento XML se transmite en un flujo de datos donde se va solicitando el siguiente evento (*Pull*) con el fin de optimizar recursos de memoria. Se distingue entre Streaming Pull Parsing donde el cliente sólo obtiene los datos solicitados previamente (SAX) y el Streaming Push cuando el analizador envía al cliente datos del XML al localizar un elemento.

✓ **JAXP** (en inglés *Java API for XML Processing*). API de Java (javax.xml.parsers y javax.xml.transform) que proporciona acceso a través de dos factorías abstractas para trabajar con instancias de analizadores DOM y SAX a través de diferentes implementaciones, así como soporte para StAX, espacios de nombres y XSLT (Xalan). También lleva incorporado el analizador Crimson. Suele integrarse en entornos con Servicios web para agrupar en un mismo *framework* todas las posibilidades de tratamiento de XML.

✓ **JAXB** (en inglés *Java Architecture for XML Binding*). API JEE (javax.xml.bind) que proporciona un conjunto de interfaces para analizar y generar XML de manera automática. A partir del modelo definido en XML realiza la generación de clases Java equivalentes. El esquema suele definirse vía DTD, a partir del cual un desarrollador puede construir un árbol de objetos Java que se corresponden con el XML. De este modo se evitan las limitaciones de memoria de DOM.

Paralelamente se dispondrá de componentes específicos para entornos basados en **Servicios web** como WSDL, los más habituales serían:

✓ **SAAJ.** API (javax.xml.soap) de SOAP y SOAP con aportaciones (en inglés SOAP *with Attachments*) que permite enviar documentos XML y aportaciones en formato MIME que pueden ser o no XML. Se acostumbra a emplear a bajo nivel por otras API para operaciones de mensajería.

✓ **JAX-RPC**. API de JEE para facilitar el desarrollo de componentes software que hagan uso de XML para comunicaciones a través de llamadas a procedimientos remotos (RPC), en la línea de IDL-CORBA y RMI. A diferencia de estas alternativas JAX-RPC emplea XML como soporte a Servicios web. Permite correspondencia entre objetos y

estructuras XML. En arquitecturas SOA el JAX-RPC sería la tecnología a través de la que el cliente envía la petición de servicio. Por debajo emplea SOAP, pero este nivel permanece transparente a la API. Sus funciones abarcan: Mensajería asíncrona, Enrutamiento de mensajes, Mensajería con entrega garantizada.

- ✓ **JAXR** (en inglés *Java API for XML Registries*). API de JEE para acceso a registros de servicios en estándares abiertos como ebXML o UDDI. Permite a los servicios la posibilidad de auto-registrarse. Asimismo soporta el uso de consultas SQL para la búsqueda de registro a través del objeto SQLQueryManager. Hace uso de JAXM para mensajería.
- ✓ **JAX-WS** (en inglés *Java API for XML Web Services*). Componente del servicio web base Metro, que sería evolución y ampliación de JAX-RPC y se encontraría integrado con JEE (javax.xml.ws). Hace uso de anotaciones Java para describir elementos de las clases, como metadatos y permiten automatización de tareas.

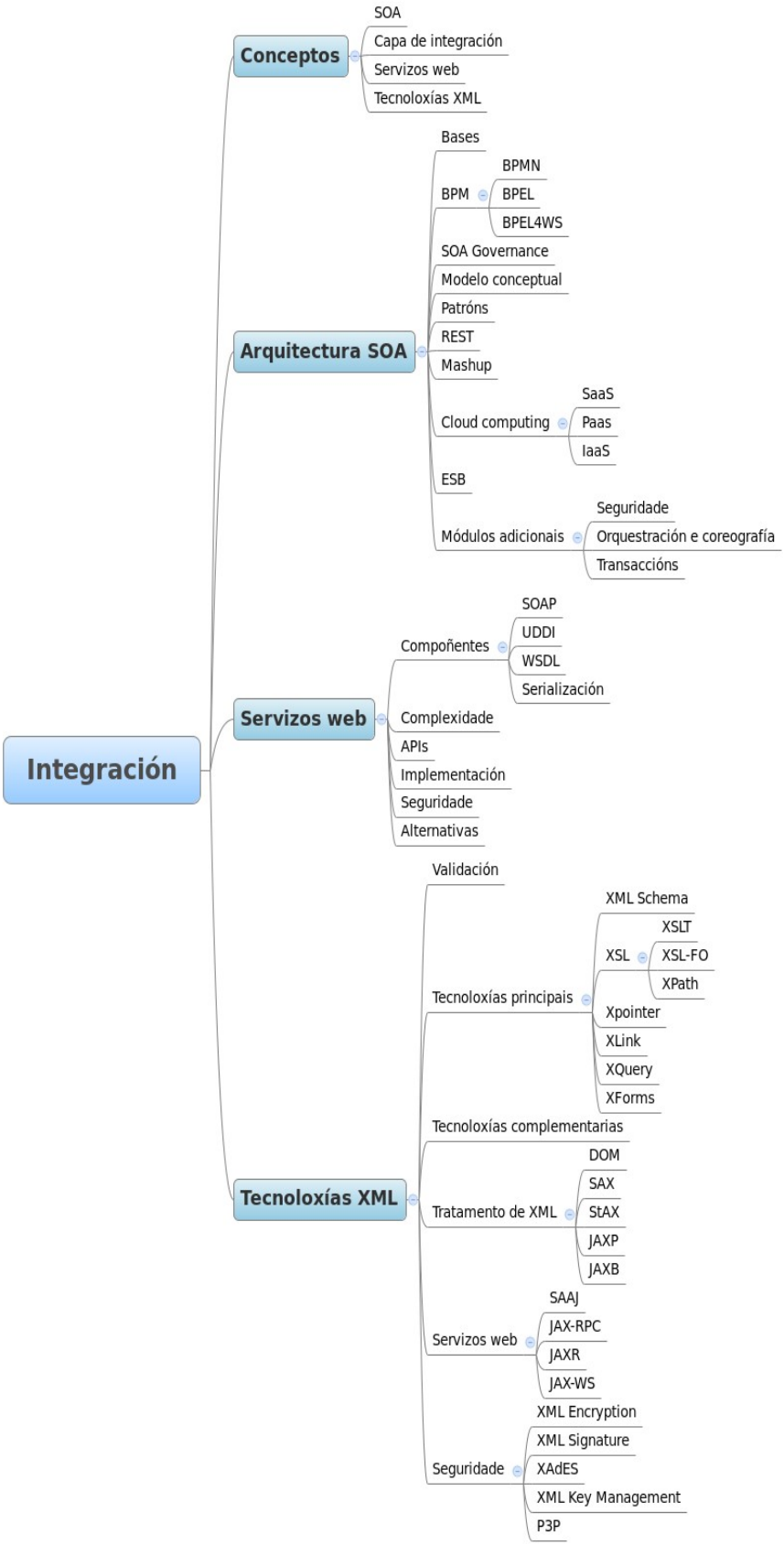
Por último dentro del ámbito de la **seguridad**, existen una serie de soluciones derivadas de la capacidad insuficiente ante arquitecturas SOA de TLS o SSL. Así hay que destacar las siguientes tecnologías:

- ✓ **XML Encryption.** La encriptación XML es un consejo del W3C que especifica el proceso para cifrar datos o documentos completos y representar esa información encriptada en un documento XML. Permite supercifrado y soporta los algoritmos TripleDES, AES y RSA.
- ✓ **XML Signature.** Firma digital que garantiza la integridad de las partes en una comunicación. Asimismo proporciona autenticación de mensajes, integridad de datos, soporte de transacciones sin repudio y firmas para cualquier contenido digital o XML. En el documento se añade un elemento Signature que encapsula el contenido de la firma

digital incluyendo una referencia al objeto firmado, la indicación del algoritmo de canonización, y el valor resultante de la firma.

- ✓ **XAdES** (en inglés *XML Advanced Electronic Signatures*). Firma digital avanzada XML, que añade un conjunto de extensiones a XML Signature, permitiendo por ejemplo que las firmas sean válidas durante largos períodos de tiempo
- ✓ **XML Key Management.** Protocolo para distribuir y registrar llaves públicas y certificados evitando la complejidad de PKI. Está compuesto de dos partes: X-KRSS o registro de llave pública, un conjunto de protocolos que soportan el registro de pares de llaves; y X-KISS información de llave pública, que define un conjunto de protocolos para procesamiento y envío de información asociada en identificada con XML Signature y cifrada con XML Encryption.
- ✓ **P3P** (en inglés *Platform for Privacy Preferences*). Especificación del W3C que define un estándar de gestión de datos y de privacidad, así como un formato XML para expresar políticas de privacidad, con el objetivo de permitir a los usuarios si y cómo quieren revelar información personal.

30.4. ESQUEMA


Integración
Conceptos
SOA
Capa de integración
Servizos web
Tecnoloxías XML
Arquitectura SOA
Bases
BPM
BPMN
BPEL
BPEL4WS
SOA Governance
Modelo conceptual
Patróns
REST
Mashup
Cloud computing
SaaS
Paas
IaaS
ESB
Módulos adicionais
Seguridade
Orquestración e coreografía
Transacción
Servizos web
Compoñentes
SOAP
UDDI
WSDL
Serialización
Complexidade
APIs
Implementación
Seguridade
Alternativas
Tecnoloxías XML
Validación
Tecnoloxías principais
XML Schema
XSL
XSLT
XSL-FO
XPath
Xpointer
XLink
XQuery
XForms
Tecnoloxías complementarias
Tratamento de XML
DOM
SAX
StAX
JAXP
JAXB
Servizos web
SAAJ
JAX-RPC
JAXR
JAX-WS
Seguridade
XML Encryption
XML Signature
XAdES
XML Key Management
P3P

TRADUCCIÓN FIGURAS:

FIGURA 1: ARQUITECTURA SOA

- Seguridad
- Mensajería
- Otros protocolos
- Otros servicios
- Descripción, interfaz

FIGURA 2: MODELO CONCEPTUAL SOA

- Servicios
- Componentes de negocio
- Sistemas operacionales

FIGURA 3: BUS DE SERVICIOS EMPRESARIALES (ESB)

- Conexiones punto a punto
- Conexiones a través de ESB

FIGURA 4: TECNOLOGÍA DE SERVICIOS WEB

- Capa de descubrimiento
- Capa de servicio. Servicios web
- Seguridad
    1. firma digital
    2. gestión
- SOA
    1. transacciones
    2. orquestación
- Tecnologías de servicios web

30.4 ESQUEMA

- Servicios web. Tecnologías XML. Patrones. Módulos adicionales. Seguridad. Orquestación y coreografía. Transacciones.
- Servicios web
    1. Componentes

2. Complejidad
3. Seguridad

- Tecnologías XML
  1. Tecnologías principales
  2. Tecnologías complementarias
  3. Servicios web
  4. Seguridad

30.5. REFERENCIAS

Varios autores.
Web Services Architecture. W3C Working Group. (2004).

César de la Torre y Roberto González.
Arquitectura SOA con tecnología Microsoft. Buenas prácticas y diseño de aplicaciones empresariales. (2008).

Joan Ribas Lequerica.
Web Services. (2003).

Patrick Cauldwell y otros.
Servicios Web XML. (2002).

Autor: Juan Marcos Filgueira Gomis
Asesor Técnico Consellería de Educación e O. U.
Colegiado del CPEIG