TEMARIO OPOSICIÓNS INFORMÁTICA

GRUPO A1 - ESCALA DE SISTEMAS E TECNOLOXÍA DA INFORMACIÓN

TEMA 30. ARQUITECTURA SOA. SERVIZOS WEB. TECNOLOXÍAS XML.

Autor: Juan Marcos Filgueira Gomis

TEMA 30. ARQUITECTURA SOA. SERVIZOS WEB. TECNOLOXÍAS XML.

30.1. INTRODUCIÓN E CONCEPTOS

Os sistemas actuais teñen unha grande complexidade debido á integración de múltiples compoñentes heteroxéneos. A comunicación e relación entre estes compoñentes é un dos grandes problemas actuais sendo **SOA** (en inglés *Service Oriented Architecture*) un dos actuais modelos de solución. Esta arquitectura entende a comunicación entre aplicacións e compoñentes coma servizos, non necesariamente servizos web, demandados por clientes ou subscritores e proporcionados e publicados por provedores. As arquitecturas para servidores de aplicacións de uso máis estendido coma .NET e JEE acostuman a definir unha **capa de integración** que agrupa os compoñentes encargados do acceso a datos, sistemas *legacy*, motores de regras de *workflow*, acceso a LDAP, etc... Para a comunicación dos compoñentes desta capa existen varias solucións coma JCA, JMS e servizos web, está última unha das máis aceptadas actualmente.

Os **Servizos web** permiten a comunicación entre sistemas heteroxéneos a través do acceso á URL de aplicacións empregando protocolos baseados en XML como SOAP ou SAAJ. Os clientes acceden ao servizo a partir da súa interface definida perante WSDL (en inglés *Web Service Definition Language*) ou inserida nalgún rexistro de servizos web.

O uso de XML convértese nun estándar de integración, sendo a base das comunicacións nesta capa, tanto para estruturar coma para almacenar e intercambiar información, estendendo o seu uso a outros ámbitos. As **tecnoloxías XML** son un conxunto de módulos que ofrecen servizos como: XSL/XSLT para deseño de documentos, Xpath como linguaxe de rutas para acceso a documentos, a linguaxe de consulta XQL, e outros como XLink ou XPointer.

30.2 ARQUITECTURA SOA

SOA define unha arquitectura orientada a servizos que busca simplificar o modelo de integración de sistemas distribuídos heteroxéneos. Nesta arquitectura os compoñentes publican e invocan servizos na rede a través de mecanismos de comunicación coma JCA, JMS, SOAP, RPC ou Servizos web. Os servizos son funcionalidades da lóxica de negocio que poden invocarse de xeito remoto para obter un resultado. Defínense perante unha interface explícita, por exemplo a través de WSDL, independente da súa implementación empregando estándares de comunicación baseados en XML. SOA define tres **bases** fundamentais:

1) **Orientación ao intercambio de mensaxes.** A base do sistema é a comunicación entre os nodos do sistema.
2) **Abstracción de compoñentes.** Cada sistema redúcese á súa interface e o conxunto de servizos que define, co cal permite a integración entre calquera tipo de sistema.
3) **Metadatos.** Descricións e información asociada a servizos e mensaxes, mellorando as capacidade semántica do sistema.

A nivel lóxico os principais compoñentes nunha arquitectura SOA son:

a) **Servizos**. Entidades ou funcionalidades lóxicas definidos en interfaces públicas, que poden ou non requirir autenticación.
b) **Provedor de servizos**. Compoñente software que implementa un servizo e publica a súa interface.
c) **Cliente de servizos**. Compoñente software que invoca un servizo dun provedor.
d) **Localizador de servizos**. Provedor de servizos que rexistra as interfaces e permite aos clientes buscar no rexistro e acceder á súa localización.
e) **Servizo de interconexión**. Provedor que comunica solicitudes de servizo a outros provedores.

O concepto de **BPM** ou Xestión de procesos de negocio (en inglés *Business Process Management*), está moi relacionado con SOA. BPM é un modelo de xestión centrado en procesos de negocio e de como integrar as súas funcionalidades en sistemas heteroxéneos. A partir da identificación e xestión dos procesos da organización pode implantarse unha solución BPM a través dunha arquitectura SOA. Froito desta idea aparecen solucións coma:

- ✔ **BPMN**. Notación para o modelado de procesos de negocio.
- ✔ **BPEL**. Linguaxe de execución de procesos de negocio con servizos web para a orquestración de servizos. Xeralmente se realiza unha conversión de BPMN a BPEL.
- ✔ **BPEL4WS.** Linguaxe de definición e execución de procesos de negocios empregando servizos web (en inglés *Business Process Execution Language for Web Services*). BPEL4WS é resultado da converxencia de WSFL (en inglés *Web Services Flow Language*) e XLANG, permitindo compoñer Servizos web coma servizos compostos denominados Servizos de negocio.

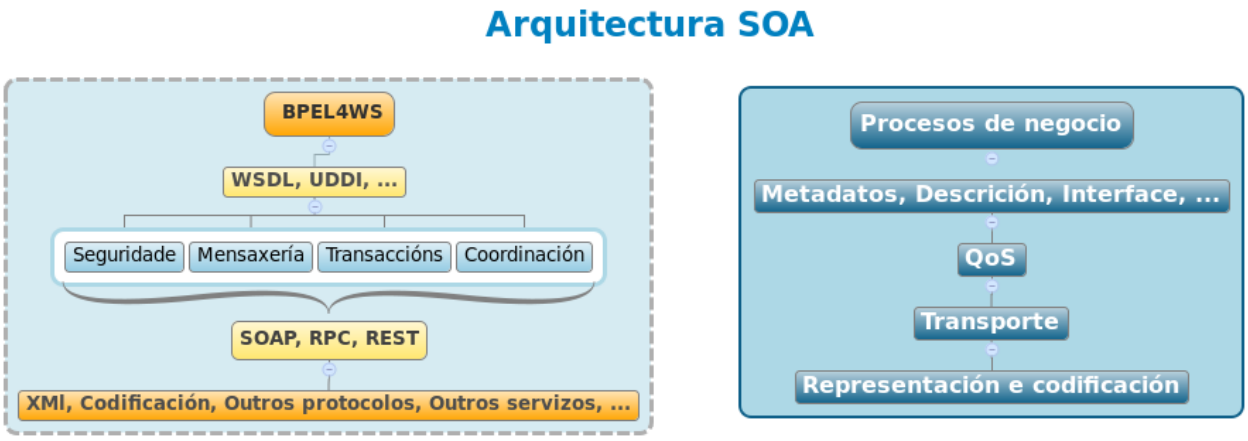


Figura 1: Arquitectura SOA.

A integración na arquitectura SOA dos BPM permite establecer o que se da en chamar **SOA Governance,** unha estrutura para a toma de decisións e establecemento de responsabilidades na organización a través da implantación de políticas, monitorización de servizos, incorporación de boas prácticas, principios arquitectónicos, mellora continua dos procesos de negocio, etc... en definitiva análise e deseño de solucións que permitan cumprir con éxito a implantación de SOA nunha organización.

A **nivel conceptual SOA** describe unha serie de guías ou patróns para servizos aliñados cun modelo de negocio. Un modelo conceptual de SOA permite definir un deseño en múltiples capas onde os servizos se relacionan con procesos de negocio e sistemas de información. A división máis habitual considera 7 capas diferenciadas, que se poden ver graficamente na Figura 2:

1) **Capa de Presentación**. Interfaces de usuario en sitios web, aplicacións e portais que invocan funcionalidades dos procesos de negocio.

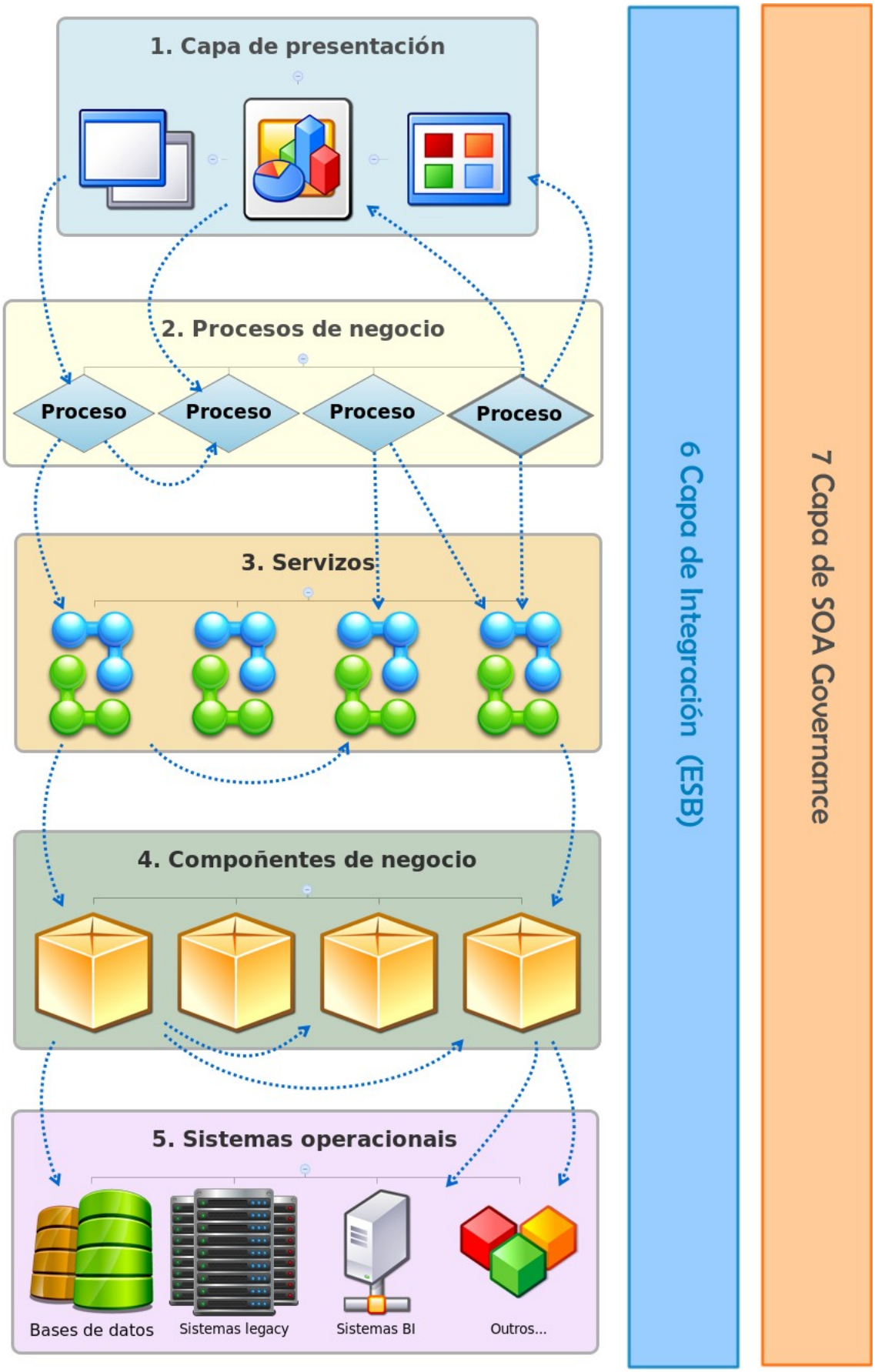


Figura 2: Modelo conceptual SOA.

2) **Capa de Procesos de negocio.** Representa os procesos e fluxos operativos ou workflows que invocan os clientes dende a capa de presentación ou orquestración de servizos. Os procesos por norma xeral representaranse con BPEL e implementarase con algunha ferramenta de transformación.
3) **Capa de Servizos.** Funcionalidades dos compoñentes da lóxica de negocio que se publican para uso dos clientes. Referéncianse a partir da interface sendo transparentes á súa implementación.
4) **Capa de Compoñentes de negocio.** Son os encargados de proporcionar as funcionalidades que se publicarán nos servizos así como calquera outra intermedia ou común ao sistema. A este nivel irían os servidores de aplicacións, outros servizos web, aplicacións, paquetes e librarías.
5) **Capa de Sistemas operacionais.** Neste nivel irían os sistemas de información da organización, sistemas *legacy*, sistemas CRM ou ERP, aplicacións de BI, etc...
6) **Capa de Integración.** Axiliza a integración de servizos a través de sistemas tipo Buses de Servizos Empresariais ou ESB, que realizan funcións de enrutamento, monitorización e administración, transformacións das mensaxes, etc... dentro da área de comunicacións.
7) **Capa de SOA Governance.** Realiza funcións de administración, monitorización e control da calidade do servizo en áreas como seguridade, dispoñibilidade e outros factores xerais non recollidos na capa de integración.

Existen **patróns de deseño** tomando como punto de partida esta arquitectura trátase de patróns para Servizos web e patróns **POSA** (en inglés *Service-Oriented Architecture Patterns*). Algúns dos principais son:

- **Service Oriented Architecture.** Patrón que define a arquitectura SOA establecendo regras, relacións e dependencias entre os compoñentes do sistema. Permite buscar servizos dinamicamente con independencia da plataforma e sen requirir implementación, con transparencia. Este patrón é unha variante ampliada do **Broker Pattern** de POSA. Neste patrón un Servizo ou nodo intermedia axuda a localizar o servizo e pode obrigar a realizar todas as comunicacións a través del ou ben unha vez establecida deixar que esta sexa directa entre o cliente e o servizo.
- **Architecture Adapter.** Patrón xenérico que facilita a comunicación entre diferentes arquitecturas grazas á independencia de usar XML/SOAP e a xeración de clases proxy. Este patrón é implementado por *frameworks* para Servizos web como Apache Axis (Java).

- **Service Directory.** Facilita a localización de Servizos web a partir dunha especificación robusta das interfaces a través do catálogo UDDI de interfaces WSDL.
- **Service Factory.** Permite a selección de servizos do provedor illando o código de comunicación UDDI. Do mesmo xeito o patrón de estendido Service Factory Cache fai funcións de caché no servizo. Simplifica en parte a API do patrón Service Directory.
- **Service Facade.** Proporciona un servizo web controlador que actúe como punto de entrada da lóxica de negocio ou obxecto de fachada. Pode empregar simultaneamente outros mecanismos de comunicación coma CORBA.
- **Event Monitor.** Emprégase para notificar que un Servizo web de longa duración invocado remotamente completa a solicitude. Cando o Servizo non dispón de mecanismos de notificación cómpre establecer un intermediario.
- **Business Object.** Un BO engloba un concepto do dominio, equiparable a un VO para contornos distribuídos.
- **Business Process.** Un BP engloba un proceso da lóxica de negocio, representando a xerarquía formada polas diferentes implementacións das súas funcionalidades e a interface do servizo.
- **Asynchronous Business Process.** Este patrón encárgase de xestionar a chamada e notificación de resposta ao cliente cando estas poden ser de longa duración.
- **Business Object Collection.** Agrupa diferentes procesos de negocio nun mesmo BOC.
- **Observer Services.** Basease nun rexistro de servizos onde o observador notifica ao cliente sobre eventos que derivados dos servizos nos que esta rexistrado.
- **Publish/Subscribe Services.** Evolución do patrón Observer Services incorporando un sistema de notificacións para substituír ao rexistro. Emprégase cando os servizos web non incorporan un sistema de notificación e precisan un intermediario.
- **Data Transfer Object.** Permite enviar múltiples obxectos nunha mesma chamada reducindo o número de conexións.
- **Partial Population.** Permite que os clientes seleccionen parte da información do mensaxe de resposta á solicitude de servizo buscando un mellor aproveitamento do ancho de banda.
- **Microkernel**. Separa un núcleo de funcionalidade mínimo de partes especificadas polo cliente.
- **Web Service Interface.** Proporciona unha interface que pode empregarse dende os clientes para invocar os métodos dun proxy de Servizo web xenérico en lugar de depender da clase proxy xerada a partir da WDSL.

REST (en inglés *Representation State Transfer*) representa un modelo de comunicación onde cada petición HTTP contén a información necesaria para responder á petición sen ter que almacenar o estado da sesión. En REST todo os servizos son recursos, identificados por URIs e se deseñan as súas representacións mediante XML, JSON ou microformatos. REST representa unha arquitectura SOA que non fai uso de Servizos web, SOAP nin RPC.

Actualmente dentro do marco da Web 2.0 xorde unha nova variante nas arquitecturas SOA, o concepto de **Mashup**, un sitio ou aplicación web que fai uso de contido doutras aplicacións ou servizos vía HTTP. Este contido é recuperado nun modelo de Servizos web a través da súa API pública evitando caer no Web Scraping. Para empregar os Mashups coma XML empréganse linguaxes específicos coma EMML (en inglés *Enterprise Mashups Markup Language*). As arquitecturas Mashup constan de tres compoñentes:

- ✔ **Os provedores de servizos.** Orixes de datos que publican a través dunha interface os métodos de acceso aso mesmos e permiten a súa consulta vía Atom, RSS, REST, JSON, Bases de datos ou interfaces WSDL de Servizos web.
- ✔ **Aplicación ou Servizo web Mashup.** Proporciona un novo servizo a partir da información obtida dos provedores.
- ✔ **Clientes.** Usuarios finais, ou outras aplicacións ou servizos que fan peticións ao Mashup. Nos clientes acostuman a empregarse tecnoloxías RIA do tipo de AJAX ou Comet.

Outro concepto que se pode relacionar con SOA é o da **Nube** (en inglés *Cloud Computing*). A nube fundamentase en empregar a rede Internet para publicar servizos, que poden ou non requirir identificación. Na nube todo son servizos, aplicacións, bases de datos, redes, e xestiónanse e accédense como tal. A arquitectura da nube, estruturase habitualmente en tres capas:

1) **Software como servizo** ou SaaS (en inglés *Software as a Service*). Sería o nivel máis alto, orientado aos usuarios e clientes finais. Incluiríanse as aplicacións propias e aplicacións de terceiros do estilo de Google Aps. A mesma infraestrutura do provedor serve a múltiples organizacións finais.
2) **Plataforma como servizo** ou PaaS (en inglés *Platform as a Service*). Serían a capa intermedia encargada de encapsular sistemas ou *middleware* permitindo correr aplicacións sobre elas. Exemplos deste servizo serían Google App Engine ou Windows Azure. Deste xeito unha organización externa proporciona un servizo de infraestrutura e soporte para outras organizacións.

3) **Infraestrutura como servizo** ou IaaS (en inglés *Infrastructure as a Service*). Neste caso ofrécense servidores para computación, rede, almacenamento ou bases de datos a través de diferentes técnicas como por exemplo a través de máquinas virtuais.

O **Bus de Servizos Empresariais** ou ESB (en inglés *Enterprise Service Bus*) representa outras das características de SOA, aínda que non é imprescindible nunha arquitectura deste tipo. Trátase dun compoñente que fai abstracción do sistema de mensaxes da organización a través dun sistema único para todos os elementos dun sistema SOA. O ESB proporciona funcións de transformación, adaptación, conexión e enrutamento que poden ser implementadas en SOA. Nunha arquitectura SOA cun ESB todas as aplicacións e servizos conéctanse a un punto único e central que administra as comunicacións, realizando funcións de *middleware*. O ESB constrúese sobre tecnoloxías XML, XSLT, XPath, JMS ou propias de Servizos web. Fai uso de elementos denominados Contedores de Servizos ou Brokers que fan a función de servidores de comunicacións. Entre os servizos que proporciona se atopan:

- ✔ Funcionalidades de enrutamento, fraccionamento e combinación de mensaxes baixo a base dos patróns EIP (en inglés *Enterprise Integration Pattern*).
- ✔ Funcións de supervisión e control da calidade do servizo a través de Acordo de Nivel de Servizos ou SLA dos servizos.
- ✔ Funcións de monitorización, seguridade e mediación de protocolos.

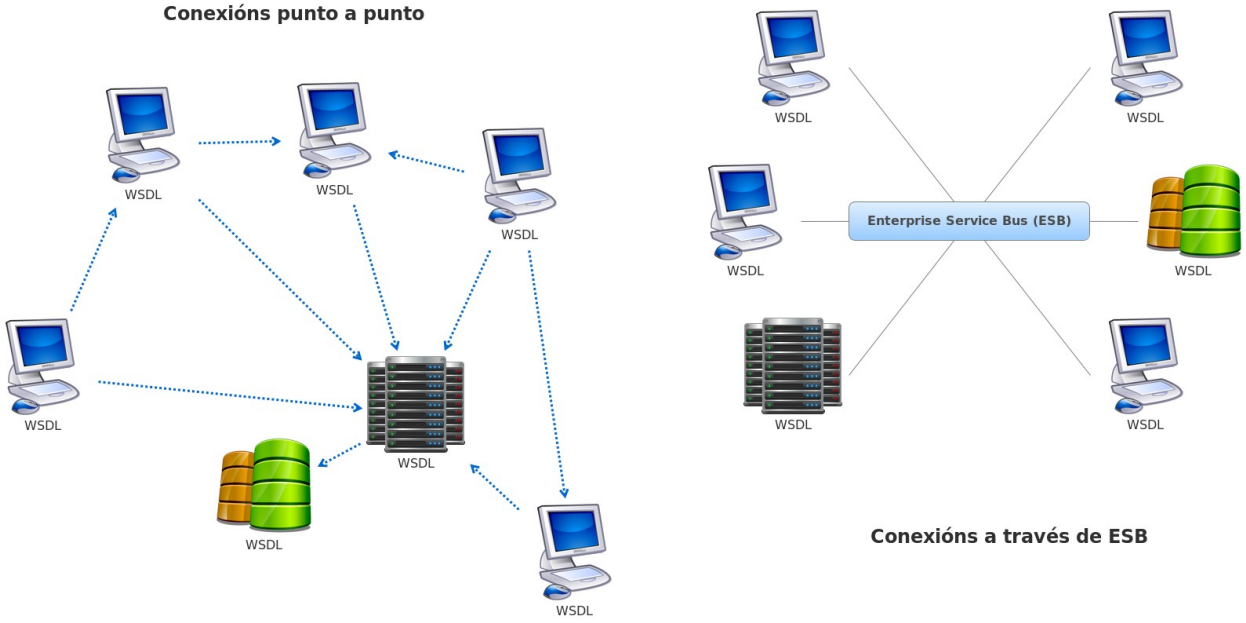


Figura 3: Bus de Servizos Empresariais (ESB).

O Bus substitúe a comunicación directa entre dous aplicacións ou servizos, de xeito que a comunicación faise de xeito transparente a través do ESB. Emprega un sistema de mensaxería, como por exemplo Tibco, soportando varios MEP ou patróns de intercambio de mensaxes, así como colas para reenviar as peticións aos provedores de servizos e as respostas ás solicitudes aos clientes. Existen *frameworks* que recollen os compoñentes precisos para implantar un ESB como Mule ESB, este é un *framework* lixeiro destinado a mensaxería e control de eventos. Permite integracións con outros *frameworks* coma Struts ou Spring e soporta moitos compoñentes de servizo coma JMS, SOAP, BPEL, JBI (en inglés Java Business Integration), e outros.

Por último sinalar que as arquitecturas SOA poden completarse con módulos específicos segundo as necesidades da organización, como:

- ✔ **Seguridade**. Coa adopción de diferentes tecnoloxías, SSL, Kerberos, X.509, Sinaturas XML, Encriptación XML, XML Canonicalization, SAML (en inglés *Security Assertion Markup Language*) ou XKMS (en inglés *XML Key Direction Specification),* que administra a chave pública ou PKI das infraestruturas.
- ✔ **Orquestración e coreografía de servizos.** Neste modelo a interacción entre servizos non se produce directamente senón que se define unha entidade que define a lóxica de interacción, facilitando a colaboración que será un servizo de control primario. En BPEL o servizo primario será un proceso BPEL, pero tamén se pode definir con BPEL4WS, WSFL ou XLANG. Mentres a orquestración precisa dun director de orquestra ou servizo central o modelo de coreografía establece interaccións punto a punto a partir de regras de colaboración xerais. Para a coreografía existen linguaxes específicas como WS-CDL (en inglés *Web Services Choreography Description Language*) que teñen definida a forma de representar as interaccións.
- ✔ **Xestión transaccional.** Existen varias tecnoloxías que coordinan as transaccións entres servizos autónomos. O BTP (en inglés Business Transaction Protocol) onde ningún dos servizos xestiona unha transacción, senón que esta se comunica a todos e deciden se se unen ou non, con comunicacións baseadas en XML nun formato propio. Outros mecanismos como WS-Transaction e WS-Coordination encárganse de xestionar transaccións levadas a cabo por varios servizos á vez, con protocolos SOAP e WSDL. Pola súa banda JEE dispón da especificación JAXTX para transaccións complexas co obxectivo de illar estas dos contedores.

30.3 SERVIZOS WEB

Os Servizos web son un dos modelos de implementación de SOA. Un Servizo web que proporciona un servizo vía web nunha rede a través dunha interface que lle permite recibir peticións e transmitir respostas. Para soportar este sistema se desenvolveron unha grande variedade de protocolos e tecnoloxías. Os principais son o HTTP/HTTPS para peticións e respostas e o XML como formato de intercambio. Os principais **compoñentes** comúns aos servizos web serían:

a) **SOAP** (en inglés *Simple Object Access Protocol*). O protocolo de comunicación, sobre a capa de transporte baseado en XML, que serve para invocar os servizos a través dun protocolo sendo os máis habituais HTTP ou SMTP, pero realmente é independente e permite outros como POP3 ou JMS. Permite tanto describir o contido da mensaxe e regras de codificación dos tipos de datos, coma aspectos de seguridade e transaccionalidade. Atópase estandarizado polo W3C, o que garante a comunicación entre sistemas heteroxéneos que o implementen.

b) **UDDI** (en inglés *Universal Description, Discovery and Integration*). Directorio onde se publican os servizos proporcionando a información necesaria para permitir a súa invocación. Presenta dúas API que permiten aos servizos publicar as súas funcionalidades e aos clientes enviar as peticións e obter os resultados. Cada servizo publícase no UDDI proporcionando a URL da súa WSDL e meta-información. De xeito xeral enténdese que UDDI proporciona tres tipos de servizos: información xeral sobre os provedores dos servizos (páxinas brancas), categorías e clasificacións de servizos (páxinas amarelas) e as regras de negocio ou información técnica sobre os servizos (páxinas verdes).

c) **WSDL** (en inglés *Web Services Description Language*). Linguaxe baseado en XML e XML Schema que permiten a descrición da interface dos Servizos web e que está estandarizado polo W3C. Nun documento WSDL defínense os tipos de datos, as mensaxes, os *endpoints*, os *bindings* e os servizos.

d) **Serialización de datos.** Empréganse definicións de XML Schema para especificar como codificar os datos en conxunción coas regras de codificación de SOAP. Aínda que o mecanismo máis habitual sexa o SOAP Document/Literal existen outros mecanismos coma: RPC/Encoding, Document/Encoding ou RPC/Literal.

Segundo o visto anteriormente para as arquitecturas SOA en xeral, pódense definir varios tipos de servizos segundo a súa complexidade, comezando polos de nivel básico aos de niveis máis complexos.

	Servizos de nivel básico	**Servizos de alta complexidade**
Función	Integración da funcionalidade dunha aplicación	Elemento chave dunha arquitectura SOA
Protocolos e tecnoloxías	SOPA, UDDI, WSDL	ebXML, BPEL, BTP, RossetaNet, Apache Axis, ...
Tipo de contido	Plano	MIME, PDF, ...
Comunicacións	Punto a punto	Multiparty, ESB, ...
Mensaxería	JMS, RPC, ...	Colaboración e *workflows*
Transaccionalidade	Non transaccional	Transaccional
Seguridade	SSL, autenticación, ...	Sinatura dixital, XML-encryption, Kerberos, ...

Táboa 1: Complexidade dos servizos web.

Segundo o tipo de comunicacións as APIs máis habituais son:

a) **API de mensaxería.** Clientes e servizos dispoñen de sistemas de mensaxería que lles permiten comunicarse en formato XML. Ao estar orientadas cara os sistemas de mensaxería presentan unha alta QoS.

b) **API de RPC.** A solución máis habitual, que emprega un compilador intermedio de WSDL para xerar o *stub* e o *skeleton* para cliente e servidor respectivamente, tal e como acontece con CORBA. Este sistema é o que acostuman empregar os *frameworks* actuais como Apache Axis.

c) **API para servidores de aplicacións (JEE/.NET).** Estas API veñen dispoñibles nas bibliotecas de clases de cada arquitectura, por exemplo en JEE disponse de: JAXM (en inglés *Java API for XML Messaging*) para intercambio de mensaxes; JAX-RPC (en inglés *Java API for XML-based RPC*) , que permite enviar peticións remotas a terceiros e recibir resultados; e JAXR (en inglés *Java API for XML Registries*), que proporciona acceso a rexistros de negocio e mecanismos para compartir información .

O **proceso de implementación dun servizo web** consiste en implementar as funcionalidades do servizo a reutilizando as clases xeradas a partir dun WSDL ou dunha API (JAX-RPC, Axis, ...). Pódense aproveitar as ferramentas existentes nos IDE, ou outras máis específicas con Ant ou *Wsdl2java.*Unha vez implementadas as clases coa lóxica do servizo xéranse as clases nun war e despréganse nun contedor de Servlets ou nun IIS. Sobre o modelo de programación empréganse diferentes variantes:

a) **Estilo CORBA.** Xéranse todas as clases ao compilar empregando clases das API (Axis, JAX-RPC, ...)
b) **Dynamic Proxy.** A interface WSDL créase ao compilar, pero o proxy no cliente só se compila en tempo de execución.
c) **Dynamic Invocation Interface.** Tanto WSDL como cliente xéranse en tempo de execución. O cliente busca e invoca o servizo vía *broker*.

Outro dos factores a considerar é o tema da **seguridade** nos Servizos web, que pola propia natureza das arquitecturas SOA resulta un tema complexo. O principais elementos de seguridade no que respecta a JEE, aínda que moitas serían extensibles a .NET serían:

- ✔ Para JAX-RPC a **API XWS-Security** que facilita a integración de aspectos de seguridade.
- ✔ O estándar **XML-DigitalSignature** para sinatura dixital.
- ✔ O estándar **XML-Encrytion** para encriptación de mensaxería.
- ✔ **Certificados X.509** para autenticación.
- ✔ Bases de datos de certificados baseadas en **JKS** (en inglés *Java Key Store*).

Dentro da arquitectura os diferentes mecanismos integraríanse nivel a nivel do seguinte xeito:

1) **Nivel de transporte.** Autenticación básica, autenticación por certificado vía SSL/TLS. Codificación de usuario/contrasinal nos *stubs* e regras de seguridade para *endpoints.*
2) **Nivel de mensaxe.** Sinatura de contidos con certificados XML-DigitalSignatura, certificados X.509 e encriptación**.**

Entre as **posibilidades** existentes para implementar Servizos web atópanse:

- ✔ APIs Java: JAX-RPC, JAXM, SAAJ (mensaxes SOAP como obxectos), JWSDL (Acceso a descricións WSDL), JAXR (Acceso ao UDDI), *framework* Apache Axis, ...
- ✔ .NET: ASP .NET, MS SOAP Toolkit, ...
- ✔ Outras tecnoloxías: NuSOAP para PHP, Axis para C++, ...

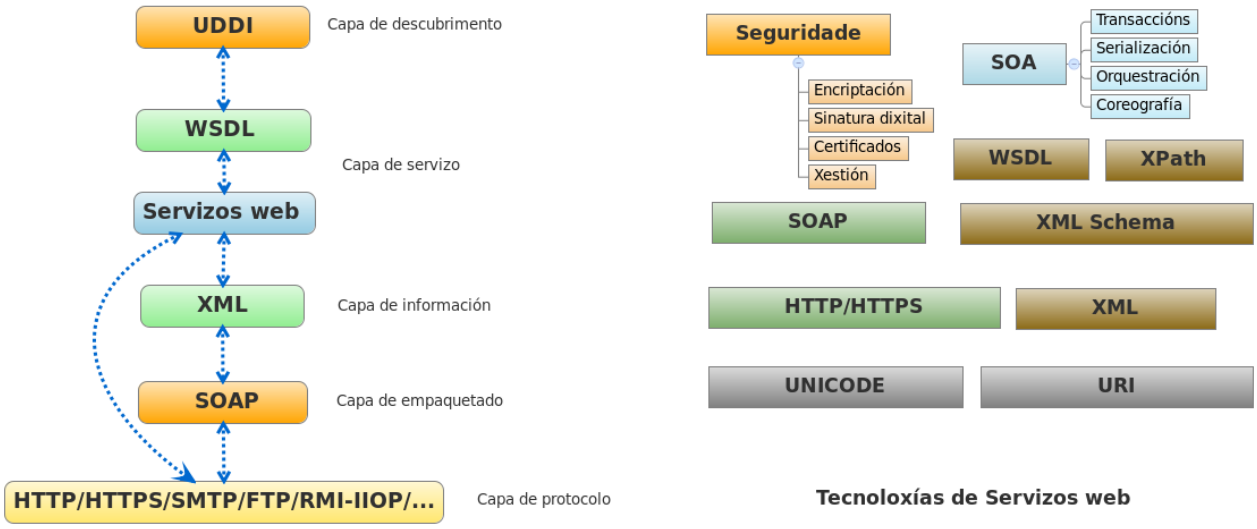


Figura 4: Tecnoloxías de Servizos web.

30.4 TECNOLOXÍAS XML

A linguaxe XML (en inglés *extensible Markup Language*) é unha metalinguaxe para etiquetado desenvolvido polo W3C. En SOA o XML representa o estándar para intercambio de información estruturada entre sistemas heteroxéneos. En esencia XML é unha linguaxe de marcas que permite a creación doutras linguaxes de marcas, con diferentes usos en SOA, cada unha destas linguaxes denomínase Aplicación XML e representa un modelo de datos de acordo a un esquema semántico.

O XML resulta máis estrito que outras linguaxes como HTML, admitindo varios mecanismos de validación ou corrección:

- ✔ **Formación**. Un documento XML denomínase "ben formado" cando segue as regras léxicas (tipo de caracteres, codificación, ...) e sintácticas (anidación correcta, marcas de apertura e peche de estrutura, ...) debidas.
- ✔ **Validación**. Un documento XML considérase "validado" cando cumpre un conxunto de regras e restricións denominadas en conxunto gramática ou Esquemas XML (en inglés XML *Schema*). Existen varios formatos para gramáticas: os DTD herdados do SGML, os XML-Schema, que son recomendación do estándar polo W3C e outros máis específicos coma o XML Data.

As **tecnoloxías principais máis empregadas** do modelo XML en arquitecturas SOA veñen dadas por:

- ✔ **XML Schema.** Linguaxe de esquema para describir a estrutura e regras de validación dun documento XML. Diferenciase do DTD en que permite un grande número de tipos de datos. Os documentos de esquema son de extensión XSD (en inglés XML Schema Definition). A programación de esquemas basease nos espazos de nomes e os elementos e atributos que conteñen. Logo de validar un documento contra un XSD pódese expresar a súa estrutura e contido en termos do modelo de datos do esquema. Esta funcionalidade denomínase PSVI (en inglés *Post Schema Validation Infoset*), e permite transformar o documento nunha xerarquía orientada a obxectos.
- ✔ **XSL**. XSL funciona como unha linguaxe avanzada para crear follas CSS transformando e realizando outras operacións sobre documentos XML, dándolles formato. Á súa vez pode descompoñerse en tres linguaxes ou dialectos XML, todos recomendacións do W3C, que integran a familia XSL:
  - ✔ **XSLT** (en inglés *Extensible Stylesheet Language Transformations*). Estándar para documentos XML que permiten transformar documentos XML en base a modelos dunha sintaxe a outra permitindo estruturas de programación, funcionando a xeito de intérprete. As regras dos modelos defínense programaticamente e a principal capacidade desta linguaxe é que permite separar o contido da presentación, ou diferentes presentacións en documentos XML o que se adapta perfectamente aos modelos de separación en capas vistos.
  - ✔ **XSL-FO** (en inglés *Extensible Stylesheet Language Formatting Objects*). Documentos XML que especifican formatos de datos ou obxectos para a súa presentación. A utilidade básica destes documentos é a presentación, co cal se complementan con XSLT na saída de datos das aplicacións. Permite a xeración de documentos multiformato: XML, (X)HTML e mesmo PDF. Existen procesadores específicos para este tipo de operacións coma Apache FOP.
  - ✔ **XPath ou XML Path Language**. Permite identificar partes dun documento XML, accedendo aos seus atributos e elementos coma se foran nodos, a través da construción de expresións que recorren e procesan un documento XML. En XSL permite seleccionar e percorrer o documento XML de entrada da transformación, pero por extensión ten outros moitos usos actualmente, servindo de base para outras linguaxes XML.

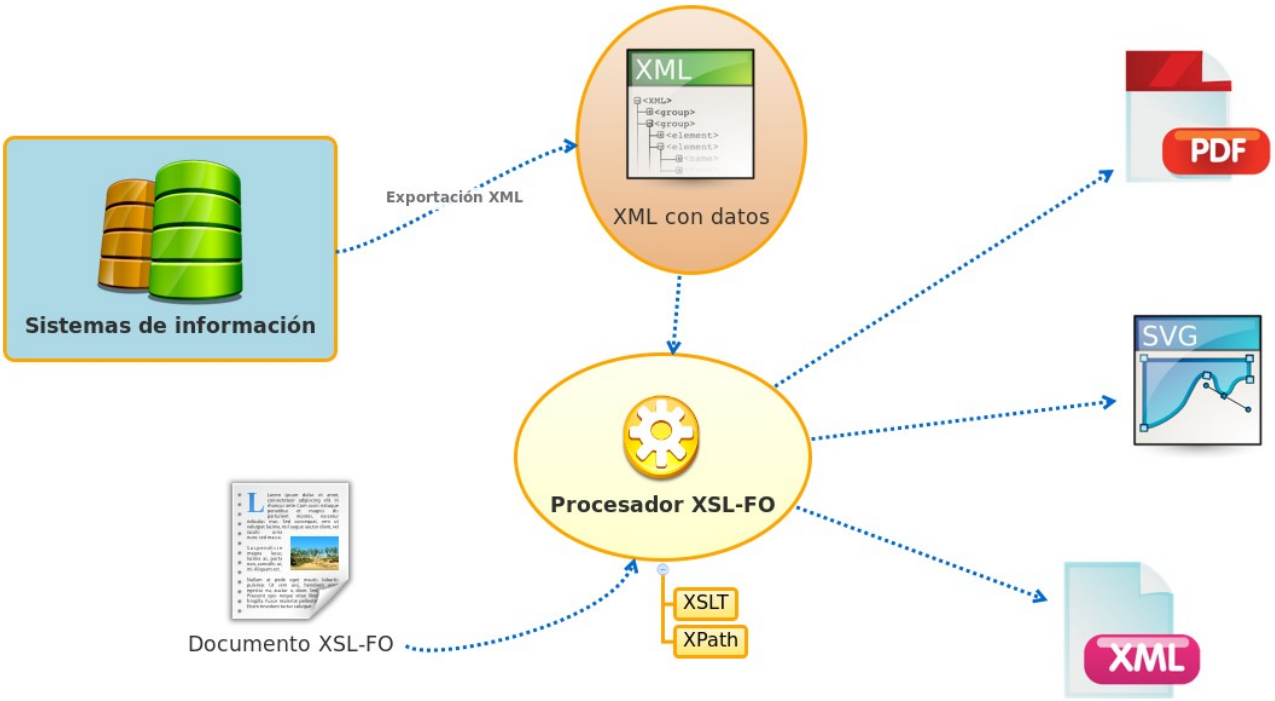


Figura 5: Familia XSL.

- ✔ **XPointer**. Recomendación do W3C que permite localizar puntos concretos ou fragmentos nun documento que expande as funcionalidades de XPath a través de rangos .
- ✔ **XLink .** Recomendación do W3C que define un mecanismo para engadir hiperligazóns en arquivos XML ou outros recursos, coa opción de navegar nos dous sentidos, con ligazóns bidireccionais ou varios arquivos enlazados, multiligazóns. En XLink todo na rede é un recurso, e se pode enlazar dende un localizador, definindo as relacións entre recursos con arcos. Así mesmo permite agregar a un vínculo información sobre si mesmo coma metadatos.
- ✔ **XQuery.** Linguaxe de consulta desenvolvido polo W3C para recuperar coleccións de datos XML, con moitas similitudes con SQL. Permite extraer e manipular información de documentos XML ou calquera outro sistema de información que permita representación vía XML como Bases de datos ou documentos ofimáticos. Emprega XPath para acceder aos documentos engadindo unhas expresións propias denominadas FLWOR. Así mesmo realiza transformacións de documentos XML e buscas de elementos textuais na web ou en recursos XML/(X)HTML. Nas arquitecturas SOA resultan especialmente útiles para recuperar información de Bases de datos e presentala a través de Servizos web.

- ✔ **XForms.** Linguaxe de definición de Interfaces de usuario desenvolvida polo W3C, centrada especialmente na parte de formularios web e a súa integración en documentos (X)HTML, ODF ou SVG. Aplícase o paradigma de separar o contido, propósito e estrutura. En aplicación do MVC incorpora un modelo declarativo de composto de regras e validación para datos e tipos de datos dos formulario, así como envío de parámetros; unha capa de vista composta dos controis da interface de usuario; un controlador para orquestrar as manipulacións de datos, interaccións entre o modelo e a vista e envíos de datos. Outras evolucións desta tecnoloxía serían AJAXForms e XSLTForms, incorporando AJAX e XSLT a esta tecnoloxía. Por outra banda, existen outras linguaxes relacionadas coas interfaces de usuario que seguen dialectos de XML, moitas delas con capacidade para interactuar con XForms como: XAML, XUL, UIML, UsiXML, AUIML, ...

O grupo de tecnoloxías anteriores poderían definirse como linguaxes XML de propósito xeral. En moitas ocasións o XML emprégase de xeito concreto para representación de datos complexos ou con necesidades específicas para o noso dominio. Na táboa 2 recóllense algúns exemplos de linguaxes XML empregadas para representación ou adaptación de información a necesidades concretas, ben para contornos de traballo como XHTML e WML ou ben en dominios específicos como aplicacións de información xeográfica ou deseño gráfico.

	Función
XHTML	HTML con especificacións máis estritas para presentar unha maior compatibilidade coa web semántica e os outros estándares XML
MathML	Expresar formulacións matemáticas
SVG	Especificación para describir gráficos vectoriais e animacións
SMIL	Permitir a integración multimedia en XHTML e SVG
WML	Adaptación do HTML para móbiles e PDA
VoiceXML	Converter fala en XML a partir de gramáticas de recoñecemento de voz
SSML	Para fala sintética
GML/KML	Para sistemas de modelado e información xeográfica
X3D	Representación de gráficos en 3D
EBML	Para almacenar xerarquías de datos en formato binario de lonxitude variable

Táboas 2: Linguaxes XML complementarias.

O uso tan estendido do XML obriga a dispoñer de ferramentas que permitan o tratamento doado dos documentos, percorrer, manipular, procesar, etc... Moitas tecnoloxías dispoñen de *frameworks* específicos para **tratamento de XML**:

- ✔ **DOM** (en inglés *Document Object Model*). Especificación do W3C dunha API (org.w3c.dom) para manipular documentos XML/HTML, acceder ao seu contido, estrutura e estilos, a través dun analizador sintáctico. DOM xera unha árbore xerárquica en memoria onde almacena todo o documento. A través dun procesador permítese acceder a calquera nodo da árbore, ou inserir/eliminar novos nodos. O principal inconveniente deste modelo é que precisa gran cantidade de memoria pola necesidade de cargar todo o documento, pero ten as vantaxes de ser moi sinxelo de implementar e de permitir a xeración de XML. Apoiase en tecnoloxías XSLT e Xpath. Frameworks como Xerces baséanse en DOM para tratamento de XML así como outros baseados en AJAX do tipo de Jquery, Prototype, Dojo, etc... Así mesmo existen alternativas recentes similares a DOM, deseñadas explicitamente para JEE que resultan máis doadas de empregar, JDOM (org.jdom) e DOM4J (org.dom4j).
- ✔ **SAX** (en inglés *Simple API for XML*). API inicialmente para Java (org.xml.sax), pero que despois evolucionou a outras linguaxes coma C++, Perl, Phyton, ... , que dispón dun analizador que xera eventos ao acadar puntos chave do documento analizado. Percorre o documento de xeito secuencial a través dun administrador de eventos, o DocumentHandler, evento a evento, co cal non precisa cargar o documento en memoria pero non permite volta atrás sen ir de novo ao inicio. Isto o fai moi axeitado para documentos de gran tamaño. Outros *frameworks* máis completos baséanse á súa vez en SAX, como: Xerces, Crimson, Piccolo ou Oracle XML Parser.
- ✔ **StAX** (en inglés *Streaming API for XML*). Define un analizador sintáctico de fluxo de datos integrado en JEE, con soporte para xeración de XML. Empréganse dous estilos de análise Cursor API e Iterador Event Iterator API, ambos baseados en iteracións para solventar as limitacións de SAX e DOM. Neste modelo o documento XML transmítese nun fluxo de datos onde o se vai solicitando o seguinte evento (*Pull*) co fin de optimizar recursos de memoria. Distínguese entre Streaming Pull Parsing onde o cliente só obtén os datos solicitados previamente (SAX) e o Streaming Push cando o analizador envía ao cliente datos do XML ao localizar un elemento.

- ✔ **JAXP** (en inglés *Java API for XML Processing*). API de Java (javax.xml.parsers e javax.xml.transform) que proporciona acceso a través de dúas factorías abstractas para traballar con instancias de analizadores DOM e SAX a través de diferentes implementacións, así como soporte para StAX, espazos de nomes e XSLT (Xalan). Tamén leva incorporado o analizador Crimson. Acostuma integrarse en contornos con Servizos web para agrupar nun mesmo *framework* todas as posibilidades de tratamento de XML.
- ✔ **JAXB** (en inglés *Java Architecture for XML Binding*) . API JEE (javax.xml.bind) que proporciona un conxunto de interfaces para analizar e xerar XML de xeito automático. A partires do modelo definido en XML realiza a xeración de clases Java equivalentes. O esquema acostuma definirse vía DTD, a partir do cal un desenvolvedor pode construír unha árbore de obxectos Java que se corresponden co XML. Deste xeito evítanse as limitacións de memoria de DOM.

Paralelamente disporase de compoñentes específicos para contornos baseados en **Servizos web** coma WSDL, os máis habituais serían:

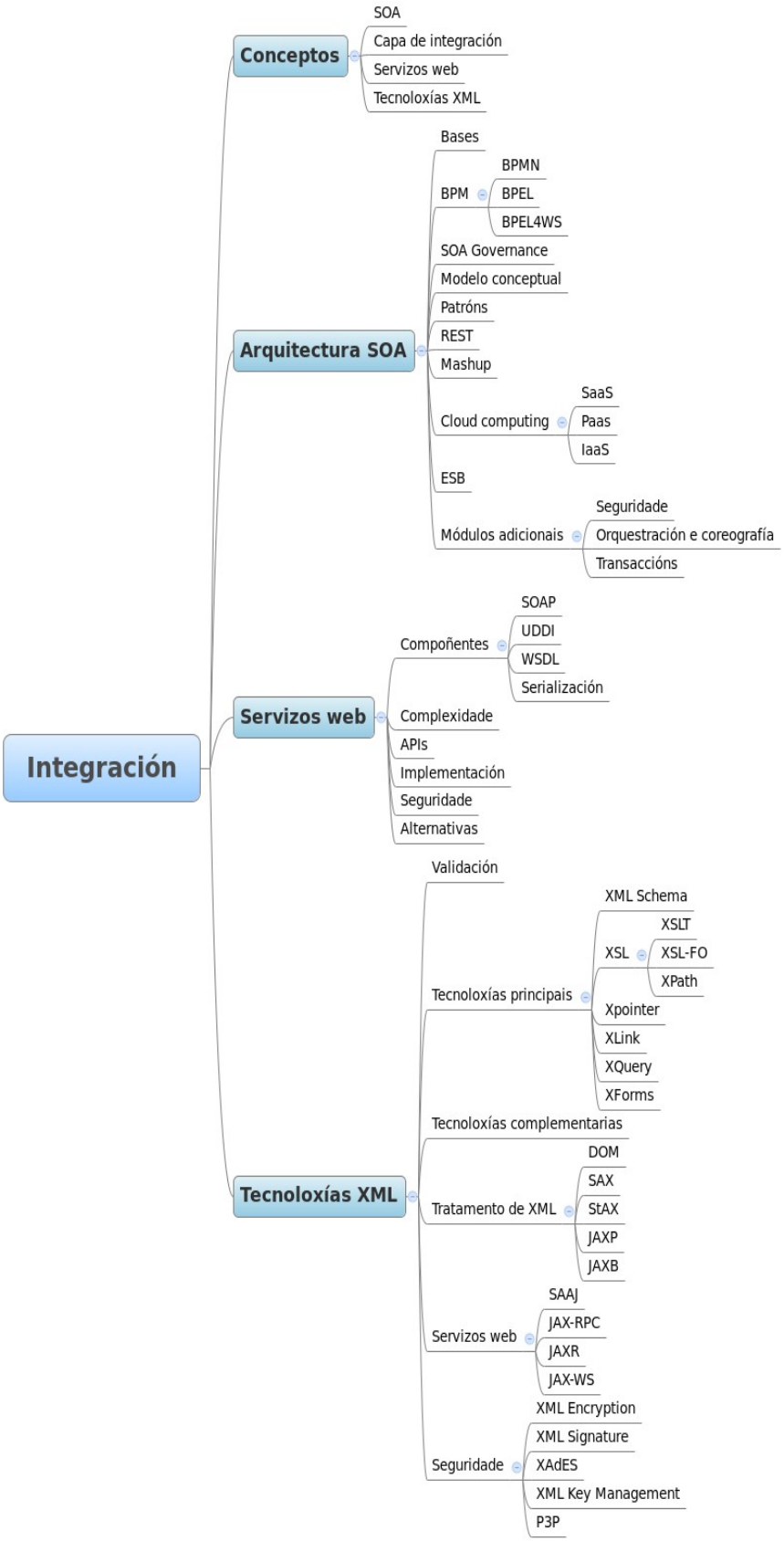
- ✔ **SAAJ.** API (javax.xml.soap) de SOAP e SOAP con achegas (en inglés SOAP *with Attachments*) que permite enviar documentos XML e achegas en formato MIME que poden ser ou non XML. Acostuma empregarse a baixo nivel por outras API para operacións de mensaxería.
- ✔ **JAX-RPC** . API de JEE para facilitar o desenvolvemento de compoñentes software que fagan uso de XML para comunicacións a través de chamadas a procedementos remotos (RPC), na liña de IDL-CORBA e RMI. A diferenza destas alternativas JAX-RPC emprega XML como soporte a Servizos web. Permite correspondencia entre obxectos e estruturas XML. En arquitecturas SOA o JAX-RPC sería a tecnoloxía a través da que o cliente envía a petición de servizo. Por debaixo emprega SOAP, pero este nivel permanece transparente á API. As súas funcións abarcan: Mensaxería asíncrona, Enrutamento de mensaxes, Mensaxería con entrega garantida.
- ✔ **JAXR** (en inglés *Java API for XML Registries*). API de JEE para acceso a rexistros de servizos en estándares abertos como ebXML ou UDDI. Permite aos servizos a posibilidade de auto-rexistrarse. Así mesmo soporta o uso de consultas SQL para a busca de rexistro a través do obxecto SQLQueryManager. Fai uso de JAXM para mensaxería.

- ✔ **JAX-WS** (en inglés *Java API for XML Web Services*). Compoñente do servizo web base Metro, que sería evolución e ampliación de JAX-RPC e se atoparía integrado con JEE (javax.xml.ws). Fai uso de anotacións Java para describir elementos das clases, como metadatos e permiten automatización de tarefas.

Por último dentro do ámbito da **seguridade**, existen unha serie de solucións derivadas da capacidade insuficiente ante arquitecturas SOA de TLS ou SSL. Deste xeito cómpre destacar as seguintes tecnoloxías:

- ✔ **XML Encryption.** A encriptación XML é unha recomendación do W3C que especifica o proceso para cifrar datos ou documentos completos e representar esa información encriptada nun documento XML. Permite supercifrado e soporta os algoritmos TripleDES, AES e RSA.
- ✔ **XML Signature.** Sinatura dixital que garantes a integridade das partes nunha comunicación. Así mesmo proporciona autenticación de mensaxes, integridade de datos, soporte de transaccións sen repudio e sinaturas para calquera contido dixital ou XML. No documento engádese un elemento Signature que encapsula o contido da sinatura dixital incluíndo unha referencia ao obxecto asinado, a indicación do algoritmo de canonización, e o valor resultante da sinatura.
- ✔ **XAdES** (en inglés *XML Advanced Electronic Signatures*). Sinatura dixital avanzada XML, que engade un conxunto de extensións a XML Signature, permitindo por exemplo que as sinaturas sexan válidas durante longos períodos de tempo
- ✔ **XML Key Management.** Protocolo para distribuír e rexistrar chaves públicas e certificados evitando a complexidade de PKI. Está composto de dúas partes: X-KRSS ou rexistro de chave pública, un conxunto de protocolos que soportan o rexistro de pares de chaves; e X-KISS información de chave pública, que define un conxunto de protocolos para procesamento e envío de información asociada en identificada con XML Signature e cifrada con XML Encryption.
- ✔ **P3P** (en inglés *Platform for Privacy Preferences*). Especificación do W3C que define un estándar de xestión de datos e de privacidade, así como un formato XML para expresar políticas de privacidade, co obxectivo de permitir aos usuarios se e como queren revelar información persoal.

30.4. ESQUEMA



30.5. REFERENCIAS

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

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

Joan Ribas Lequerica.
Web Services. (2003).

Patrick Cauldwell e outros.
Servicios Web XML. (2002).