Tabla de Contenidos

1 Introducción a XML

Al confrontar la gran variedad de formatos en que sus datos están almacenados, muchos usuarios se preguntan porqué la industria informática no utiliza un formato universal que permita intercambiar fácilmente información entre aplicaciones. Evidentemente, enunciar ese objetivo es mucho más fácil que alcanzarlo. De hecho, esta inquietud ha existido desde hace tiempo. En el pasado, sin embargo, no se había podido alcanzar un mínimo de acuerdo que permitiera un avance significativo en ese campo.

Por otro lado, en la industria de Internet existía la preocupación de que las tecnologías existentes en ese momento retrasaban la evolución de Internet ya que resultaban insuficientes para la administración de sitios y el desarrollo de aplicaciones revolucionarias. Entre las limitaciones más importantes se pueden mencionar las siguientes: las aplicaciones Web estaban restringidas principalmente a clientes que se ejecutaban en navegadores; las aplicaciones del lado del servidor no consideraban la posibilidad de comunicarse con agentes programados o con otras aplicaciones; todo el contenido se envíaba como HTML, lo que limitaba la capacidad de procesamiento posterior o cambio en el formato por parte del cliente; y, finalmente, la comunicación cliente-servidor era principalemente sincrónica, esto es, tanto el cliente como el servidor debían coincidir en el tiempo.

Las inquietudes anteriores impulsaron una iniciativa para la definición de un lenguaje estándar que por un lado facilitara el intercambio de datos entre aplicaciones y, por otro, permitiera desarrollar aplicaciones más poderosas en Internet. El lenguaje desarrollado, XML, está basado en un estándar previo llamado SGML[1], el cual, aunque muy poderoso, resulta difícil de usar. En el desarrollo de la iniciativa XML han participado múltiples organizaciones que forman parte del Consorcio del Web[2] (W3C): entre ellas Adobe, Fuji Xerox, Hewlett-Packard, IBM, Microsoft, Netscape, Sun Microsystems; vendedores de SGML e integradores de sistemas como ArborText, Inso, SoftQuad, Grif, Textel e Isogen; y la comunidad académica con NCSA y Text Encoding Initiative. Además de estar basado en las características y alcances de SGML, el lenguaje XML debía tener la sencillez y difusión de HTML.

SGML es un estándar muy amplio que permite describir con gran profundidad el contenido lógico de múltiples clases de documentos. Tiene muchas opciones que le permiten modelar tipos de documentos muy complejos y permitir su procesamiento automático. Por un lado, esa amplitud y flexibilidad es ventajosa porque permite manejar situaciones difíciles; como por ejemplo, documentos con delimitadores opcionales, documentos con partes obligatorias pero en orden arbitrario. Un caso muy claro de ese poder está en la capacidad de SGML para modelar documentos que deben ser analizados desde varias perspectivas distintas a la vez: para un mismo documento se tiene su estructura lógica (capítulos, secciones, etc.) y su organización física página por página). Por otro lado, este gran poder descriptivo, también es fuente de la complejidad que forma la mayor desventaja de SGML. Los sistemas que lo implementan en su

[1] SGML: Standard Generalized Markup Language (ISO8879, se puede adquirir en el sitio de ISO URL: http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=16387). Es un meta-lenguaje, es decir, un lenguaje para definir otros lenguajes de etiquetas.

[2] W3C: World Wide Web Consortium (http://www.w3.org/), creado en 1994 para llevar a Web a su máximo potencial desarrollando protocolos que promuevan su evolución y aseguren su interoperabilidad.

totalidad son muy complejos y caros. Durante el diseño de XML se reconoció la importancia de que el estándar propuesto fuera lo más simple posible con el fin de ampliar su base de usuarios. Toda la experiencia previa provista por el SGML representaba un punto de inicio muy sólido para el nuevo estándar. Con el fin de simplificarlo, sin embargo, se eliminaron aquellas características de SGML que aunque poderosas, no correspondían directamente al ámbito de acción del XML. Esto ha permitido que las herramientas que manejan XML sean significativamente más fáciles de desarrollar que las herramientas que manejan la totalidad de SGML.

1.1 Orígenes y motivación

En 1996 el W3C comenzó el proceso de diseñar un lenguaje que combinara la flexibilidad y el poder de SGML con la aceptación generalizada del HTML[3]. El objetivo en sí era diseñar un nuevo lenguaje totalmente compatible con SGML que fuera mucho más fácil de usar y más barato de desarrollar que SGML. El responsable de dirigir los esfuerzos del Grupo de Trabajo del W3C fue Jon Bosak (Sun Microsystems), quien apreciaba el poder y la flexibilidad de SGML y también tenía la convicción de que HTML no satisfacía los requerimientos de la siguiente generación de Internet. El producto de este proceso fue la recomendación de XML del W3C [W3C 1998], en la que se detallan los objetivos de diseño de XML:

- abiertamente utilizable en Internet.
- deberá soportar una amplia variedad de aplicaciones.
- compatible con SGML.
- deberá resultar sencillo escribir programas que procesen documentos XML.
- el número de características opcionales de XML deberá mantenerse en el mínimo absoluto, idealmente cero.
- los documentos XML deben ser legibles para el usuario y razonablemente claros.
- el diseño de XML debe prepararse lo más rápido posible, además de ser formal y conciso.
- los documentos XML deben ser fáciles de crear.
- la concisión en las etiquetas de XML (longitud) tiene importancia mínima.

Desde el punto de vista de los usuarios, estos tenían básicamente las siguientes necesidades [Bosak 1998]:

- **Extensibilidad**, no debe haber un conjunto de etiquetas rígido como el de HTML sino que debe ser posible definir nuevas etiquetas cuando sea necesario.
- **Estructura**, modelar los datos con cualquier nivel de complejidad.
- **Validación**, comprobar fácilmente la corrección estructural de los datos.
- **Independencia del medio**, poder publicar contenidos en una gran variedad de formatos.
- **Independencia de plataforma y vendedor**, debe ser posible procesar cualquier documento válido utilizando un software comercial o cualquier herramienta de texto sencilla.

[3] HTML: HyperText Markup Language (ISO/IEC 15445, URL: http://www.cs.tcd.ie/15445/15445.htm), lenguaje particular de etiquetas para el formateo de texto en navegadores Web.

En 1998 el W3C aprobó la Recomendación de XML 1.0, en la que se describe una clase de objetos llamados documentos XML y se describe parcialmente el comportamiento de los programas de computadoras que los procesan.

Esta especificación, junto con los estándares asociados (Unicode e ISO/IEC 10646 para caracteres, Internet RFC 1766 para etiquetas de identificación de lenguajes, ISO639 para códigos de nombres de lenguajes, e ISO 3166 para códigos de nombres de países), proveen toda la información necesaria para entender XML y construir programas que lo procesen [W3C 1998].

1.2 Estado actual[4]

Para demostrar el impacto que ha tenido XML se utilizará como ejemplo la industria de Internet. Es allí donde probablemente se han desarrollado la mayor cantidad de estándares y se ha podido valorar el impacto real, su uso y aplicación. Dentro de esta industria, la entrega de contenidos y la integración de aplicaciones presentan una mayor actividad y partipación de XML.

Normalmente, la entrega de contenidos puede involucrar que estos sean presentados a los usuarios finales en forma electrónica, o empacados en forma física para el envío eventual a los usuarios. Un sistema de entrega debe soportar todos los formatos de entrega requeridos, como websites, FTP, correo electrónico, o sindicación a otros websites, como proveedores de noticias. Esto requiere que el repositorio soporte el almacenamiento de contenidos en formatos XML (probablemente en varios idiomas), para facilitar la generación automática del contenido en múltiples formatos como, HTML para navegadores de Internet o WML para teléfonos WAP. Estos sistemas, además, tienen requerimientos de integración que las tecnologías estándar basadas en Internet suelen satisfacer. XML se ha perfilado en los últimos años como la base de tales arquitecturas no solo por sus aportes en el intercambio de información sino por el surgimiento de importantes lenguajes derivados de XML para administrar todos los aspectos de su estructura y procesamiento.

Por ejemplo, es común en desarrollos basados en capas, que la capa intermedia requiera funcionalidades de transformación que cambien el contenido o formato del contenido de un mensaje, dependiendo de las aplicaciones fuente y destino involucradas. Dado que la mayoría de las aplicaciones entienden los datos en una variedad de formatos incompatibles, estas transformaciones aseguran que cada aplicación puede procesar exitosamente el mensaje recibido. XML es el mecanismo estándar de facto para la transformación de datos, por lo que el software de transformación debe soportar XML.

Las soluciones de integración externa de aplicaciones B2B (de negocio a negocio) emplean XML de varias maneras. Cada aplicación, internamente, utiliza formatos propietarios de documentos basados en XML para codificar la información de sus transacciones. Para lograr la integración, estas aplicaciones cuentan con mecanismos de conexión y comunicación, también basados en XML que administran el flujo de datos entre las mismas, desde el transporte sobre la red hasta la transformación de los datos en formatos auxiliares o intermedios.

Dado que XML tiene asociadas ventajas como poco riesgo, simplicidad, apertura, extensibilidad,

[4] A setiembre del 2004.

bajo costo y facilidad de implementación, se justifica que sea considerado el estándar de datos preferido para la integración de datos. Por ello, muchos consorcios de industrias han definido procesos de negocio comunes y estructuras de datos utilizando las especificaciones de XML disponibles en forma gratuita. Además, aprovechan que XML es extensible para definir estándares de documentos XML a la medida de sus requerimientos y, así, integrarse con un amplio rango de proveedores.

Por último, las aplicaciones de negocios sobre Internet dinámicas se concentran en ubicar y disponer en tiempo real los procesos de los negocios de manera tal que respondan más eficientemente a las demandas cambiantes de los clientes y de los mismos negocios. Estos sistemas se basan principalmente en la tecnología de Servicios Web (Web Services), para acceder y emplear software desarrollado por entidades externas. Esta a su vez, consiste en un conjunto de estándares basados en XML: UDDI, SOAP y WSDL[5]. Los procesos de búsqueda, descubrimiento y descripción de servicios siguen el estándar UDDI. El estándar SOAP se utiliza para proveer la mensajería requerida para conectar componentes de aplicaciones sobre Internet, también basada en XML. Finalmente, el desarrollo de aplicaciones con Servicios Web procura la creación de software como un conjunto interconectado de componentes de software, envueltos en WSDL, el cual provee los elementos para completar la funcionalidad de la aplicación.

Como se ha mencionado anteriormente, con la rápida adopción de los estándares XML, los consorcios de importantes industrias comenzaron a crear esquemas específicos para estructurar los datos XML y así favorecer la integración de sus miembros. De este modo han surgido un gran número de vocabularios en áreas como finanzas, bancos, seguros y salud. El W3C ha especificado entre enero del 2000 y setiembre del 2004, más de 896 recomendaciones, de las cuales 72 corresponden a publicaciones, 67 son propuestas, 79 son candidatas, 87 llamadas a revisión y 672 trabajos publicados relacionados y notas. La mayoría de estos estándares corresponden a vocabularios verticales específicos de ciertas industrias, otro buen porcentaje resuelve operaciones comunes entre grupos de industrias y en menor número especifican como almacenar, transmitir, encriptar (cifrar), recuperar y transformar XML.

Por ejemplo, algunos de los estándares XML específicos para los negocios sobre Internet son:

- **RosettaNet**: Define un protocolo de negocios que permite a las empresas conducir su negocio sobre Internet. El protocolo comprende un conjunto de estándares y procesos para la automatización de cadenas de abastecimiento entre manufactureras, distribuidores y revendedores dentro de las industrias de tecnologías de información, y fabricación de semiconductores y componentes electrónicos. Fue propuesto por la organización RosettaNet, subsidiaria de UCC (Uniform Code Council, Inc.).

[5] UDDI (Universal Description, Discovery, and Integration): un registro de servicios Web junto con un mecanismo de localización y búsqueda; almacena y categoriza información de negocios y permite obtener referencias a interfaces de servicios Web.
SOAP (Simple Object Access Protocol): define por medio de XML una envoltura de comunicación para los servicios web que permita usar protocolos de transporte como HTTP; provee además de un formato de serialización para la transmisión de documentos y de una convención para describir interacciones tipo RPC.
WSDL (Web Services Description Language): define por medio de XML las interfaces de los servicios, los tipos de los datos y de los mensajes, y los patrones de interacción empleados.

- **ebXML**: Comprende un conjunto modular de especificaciones de negocios sobre Internet. Fue propuesto por el grupo OASIS en conjunto con la Agencia CEFACT de las Naciones Unidas.
- **tpaML**: Este estándar de IBM utiliza XML para definir e implementar contratos electrónicos.
- El software **SWIFT** de interfaces y servicios estandarizados para la industria financiera, permite para el intercambio de información bancaria en forma segura. Abarca al estándar **FIX** para datos financieros y al estándar **IFX** para transacciones financieras. Es utilizado por alrededor de 7650 industrias en 200 países.
- **OAGIS**: Es una iniciativa de definición de un lenguaje de negocio canónico para la integración de información. Comprende un conjunto de especificaciones XML y estándares que aseguran la interoperabilidad de las aplicaciones empresariales entre compañías. Recientemente se publicó una biblioteca de archivos WSDL para facilitar el desarrollo de OAGIS en ambientes de Servicios Web.
- **BPML**: Este estándar permite describir procesos de negocios (workflows). Su sucesor es el lenguaje BPEL4WS (Business Process Language for Web Services) adaptado para los ambientes de Servicios Web. Fue propuesto por BEA, IBM, Microsoft, Siebel Corporation y SAP.

Diversas organizaciones han dedicado gran cantidad de esfuerzos al diseño y desarrollo de herramientas que soporten estos estándares y faciliten a las organizaciones el éxito en su incorporación.

1.3 Limitaciones

A pesar de todos los beneficios que ofrece XML, aún existen áreas abiertas de investigación y problemas sin resolver que limitan la adopción de sus estándares en forma generalizada. Algunos de ellos son:

- El intercambio de datos requiere una concordancia en los significados semánticos de los objetos involucrados, lo que se complica ya que en algunas industrias se han multiplicado las propuestas de estándares de diccionarios XML, que en algunos casos son contradictorios, por lo que obligan a los miembros de una industria a adoptar varios estándares y a crear conversiones entre ellos.
- Las tecnologías de "middleware" orientadas a mensajería se encuentran aún en su fase de maduración, por lo que resulta arriesgado comprometerse con algunas soluciones.
- Los servicios Web provocan un alto consumo del ancho de banda de la red, faltan mecanismos de seguridad y el soporte a transacciones complejas [Watt 2002].

1.4 Características principales de los estándares XML

XML es considerado como una gran victoria de los estándares abiertos para los usuarios. En primer lugar, se puede extender libremente y no tiene limitaciones de nombres de etiquetas en el lenguaje. Además, es legible para los humanos, e incluso se pueden mantener los datos

utilizando herramientas de texto básicas como sed[6], awk[7] y Notepad. Finalmente, Perl[8] es un lenguaje de programación óptimo para el soporte de XML. De ahí que, en teoría, los usuarios de XML no tienen porqué estar sometidos al control de un proveedor particular de tecnología.

XML fue diseñado explícitamente para que fuera fácil implementar herramientas que lo manejen. Esto ha permitido la aparición rápida de herramientas comerciales, baratas y poderosas[9]. Además, también hay una creciente disponibilidad de herramientas XML libres[10], la mayoría de ellas desarrolladas en Java [Bosak 1998b].

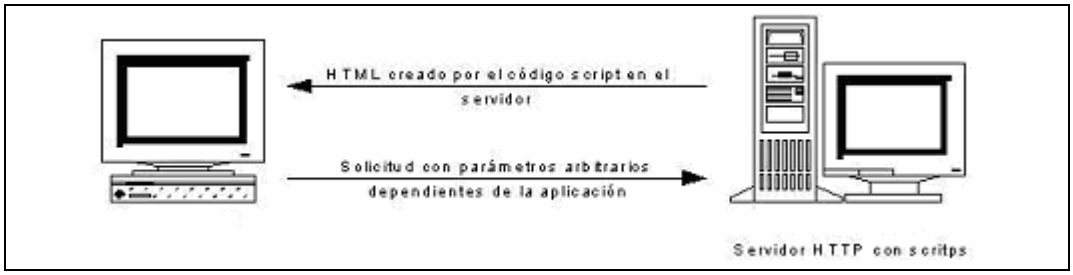
Gracias a esto, XML ha tenido una explosión importantísima en los últimos años, al punto de haber sido clasificada como uno de los diez conceptos tecnológicos más importantes del año 2001 por InfoWorld[11]. Actualmente se pueden encontrar aplicaciones XML prácticamente en cualquier dominio, las más importantes son:

- Transacciones de negocios,
- Comunicación entre procesos,
- Mensajes electrónicos,
- Tablas de bases de datos,
- Llamadas a procedimientos remotos,
- Páginas Web,
- Datos de configuración, dibujos, ecuaciones matemáticas, artículos, etc.

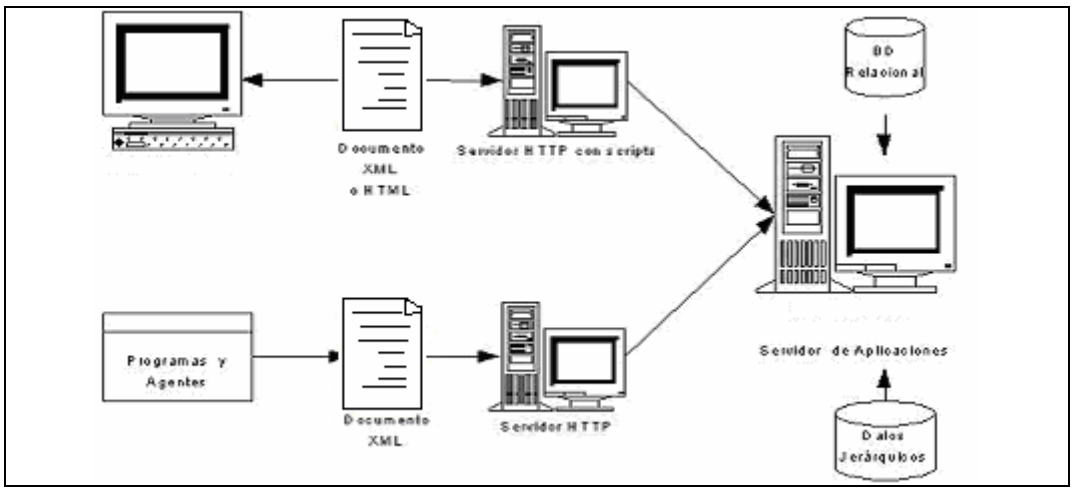
El mecanismo de descripción de datos de XML es ideal para compartir información en Internet porque es abierto, es decir, XML puede utilizarse para intercambiar datos con otros usuarios o aplicaciones independientemente de la plataforma en que se encuentren. Su naturaleza auto-descriptiva en sí misma es una característica útil para las aplicaciones negocio a negocio y las soluciones extranet. Esto también permite compartir datos entre aplicaciones sin necesidad de una coordinación previa [Martin 2000].

En la figura 1.1 se pueden distinguir los componentes básicos que diferencian la arquitectura Web clásica y la arquitectura Web basada en XML.

[6] Sed es un editor de texto para scripts de UNIX.
[7] Awk es un lenguaje de programación con grandes facilidades para el procesamiento de texto.
[8] Perl es un lenguaje de procesamiento de texto que provee facilidades muy sofisticadas, por lo que se ha popularizado para tareas como entrada y salida, administración y procesamiento de archivos, y administración de sistemas.
[9] En el Sitio "Free XML Tools and Software" se puede encontrar una recopilación importante de herramientas XML disponibles en el mercado. URL: http://www.garshol.priv.no/download/xmltools/
[10] La mayoría de las herramientas en el mercado permiten versión de prueba por tiempo limitado. Entre las más conocidas está Spyxml (http://www.spyxml.com), la cual ha ganado reconocimientos muy importantes desde que salió al mercado.
[11] Junto a XML están: Web Services, WLANs, Peer-to-peer, Sistemas de 64 bits, Portales, CRM, NAS, Bases de datos y Handhelds (URL: http://www.infoworld.com/articles/ne)



(a) Arquitectura Web Clásica



(b) Arquitectura Web basada en XML

Figura 1.1 Arquitecturas Web

En la arquitectura clásica la aplicación cliente es un navegador que actúa como lo haría una persona. Este envía una solicitud de una página a un servidor HTTP. La solicitud incluye un conjunto de parámetros y valores que son particulares para la aplicación. El servidor satisface la solicitud creando código HTML dinámico por medio de un lenguaje de "script" como JSP, CGI o ASP. Al recibir los datos en formato HTML, el cliente se ve limitado a la estructura de los datos que está inmersa entre las etiquetas HTML, llegando incluso a depender del formato y medio en que fue generada la página. En cambio, en la arquitectura basada en XML, un cliente, que puede ser un navegador o una aplicación (como es el caso de los Web Services), envía un documento XML como solicitud al servidor. La estructura de la solicitud puede especificarse utilizando mecanismos estándar que se obtienen directamente del servidor en tiempo de ejecución. Este mecanismo describe la estructura esperada por el servidor y le permite al cliente verificar si la solicitud está correcta antes de enviarla. Es de este mecanismo de comunicación del que se han apoyado los diferentes sectores de la comunidad de Internet para establecer la comunicación servidor-a-servidor y facilitar una gran cantidad de aplicaciones innovadoras que utilizan XML como mecanismo de intercambio de información.

Para construir una arquitectura Web basada en XML se requieren varias tecnologías que, como capas de cebolla, permiten alcanzar la gran variedad de objetivos que satisface XML. La primera capa consiste de las tecnologías que permiten crear datos XML y establecer mecanismos de definición, manipulación, transformación y enlace. Entre estas tecnologías se encuentran las

DTDs, Schemas, NameSpaces, XSL, XSLT, Xlink y Xpointer. En una capa intermedia, entre los datos y la aplicación, están los mecanismos de comunicación entre componentes como RPC, WIDL y WSDL. Los dos primeros permiten conectar aplicaciones B2B entre sí y con los sitios Web, a través de Internet o de una Extranet; por su lado, WSDL permite describir Web Services para su localización y uso. En una última capa se encontrarían los muchísimos vocabularios basados en XML que han sido creados para varios dominios de aplicación, junto con los mecanismos de administración y publicación de vocabularios que ha definido la W3C (NameSpaces).

En la Figura 1.2 se puede observar el flujo de transformación que involucra la generación de documentos en un ambiente basado en XML. El documento XML de la izquierda puede ser el resultado de una consulta a una base de datos que es validado por el parser de acuerdo a la DTD, el Schema o el NameSpace que tenga asociado. Luego, este archivo es transformado por el Procesador XSL, aplicando una hoja de estilo que puede contener etiquetas de los diversos estándares de transformación y enlace existentes.

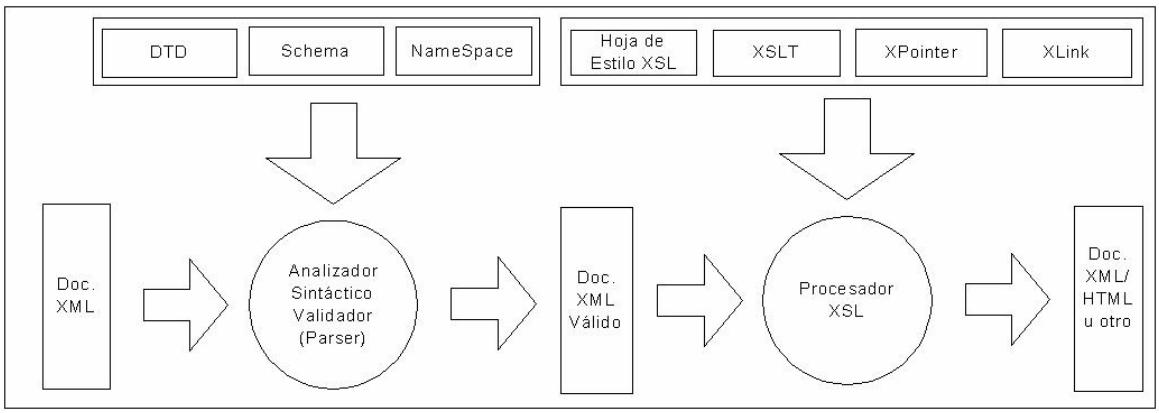


Figura 1.2. Tecnologías XML para crear, definir, manipular, transformar y enlazar documentos.

En los siguientes capítulos se presentarán con mayor detalle los principales estándares mencionados y más adelante se describirán algunas de las aplicaciones más importantes de XML como son el Comercio Electrónico y la comunicación entre procesos.

2 Sintaxis de XML

En este capítulo se describen los principales elementos sintácticos de XML. Estos elementos permitirán entender los ejemplos que se presentan en este informe y que le darán al lector la capacidad de reconocer las partes y componentes de un documento XML.

A lo largo de los próximos tres capítulos se utilizará, como ejemplo, una aplicación que resulta muy conocida, el correo electrónico. Esta aplicación, que se ilustra en la figura 2.1, permitirá mostrar algunos de los conceptos sobre documentos estructurados que rodean a XML.

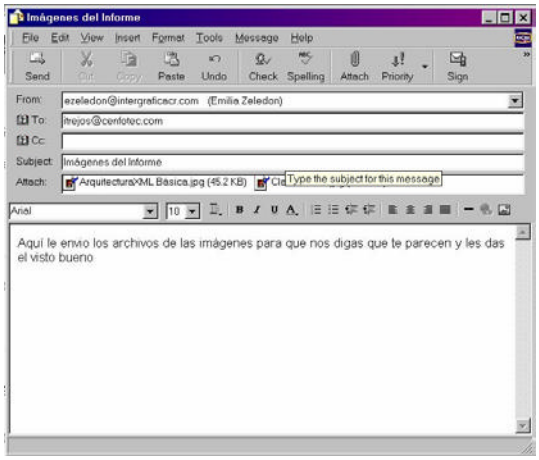


Figura 2.1 Aplicación ejemplo: correo electrónico

Para este informe, un documento estructurado es aquel cuyo contenido se puede describir por medio de una jerarquía de elementos, donde el inicio y el fin de cada elemento se marcan por medio de etiquetas. Para el caso de un documento de correo electrónico, su estructura jerárquica se muestra en la figura 2.2. Como muestra la figura, el mensaje de *correo electrónico* consta de tres partes: *encabezado*, *mensaje* y *adjunto*. El *encabezado* a su vez consta de varias partes como son *destinatario*, *remitente*, *fecha* y *asunto*.

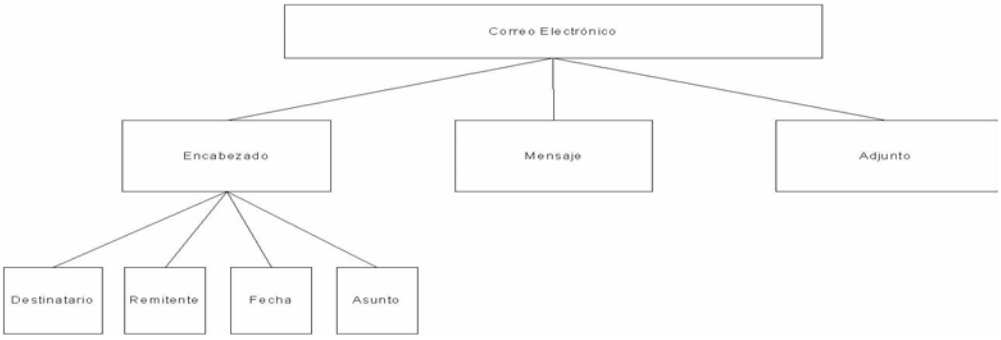


Figura 2.2 Estructura jerárquica de un documento

2.1 Etiquetas

Las etiquetas describen la estructura y el contenido del documento. La estructura de las etiquetas

de XML es esencialmente la misma de las etiquetas de HTML, tal como se muestra en la figura 2.3. Las etiquetas siguen las siguientes reglas:

- <*Etiqueta*> es la etiqueta de inicio de un elemento
- <*Etiqueta Atributo*="*Valor*"> indica una etiqueta de inicio con un atributo
- </*Etiqueta*> es la etiqueta de cierre de un elemento
- <*Etiqueta* /> es la etiqueta de un elemento vacío
- <*Etiqueta Atributo1* ="*Valor*" *Atributo2* ="*Valor*"/> es la etiqueta de un elemento vacío con atributos

Con el fin de facilitar el procesamiento de documentos XML algunas cosas que son opcionales en HTML son obligatorias en XML; por ejemplo, la etiqueta final. XML es sensible a las mayúsculas y minúsculas y utiliza como conjunto de caracteres el estándar 16-bit+ Unicode 2.1[12].

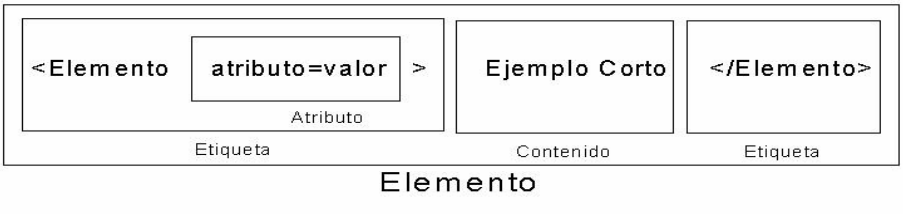


Figura 2.3 Estructura de etiquetas

2.2 Estructura de un documento

Un documento XML está compuesto de tres partes:

Encabezado o *prólogo*: es opcional, incluye información sobre la versión de XML que utiliza el documento. Puede incluir también un modelo que describe la jerarquía de elementos que forman al documento; dicho modelo puede ser incluido explícitamente dentro del documento o puede residir en un archivo externo[13].

Cuerpo del documento: incluye la estructura jerárquica con los datos que contiene el documento. Las etiquetas que se utilizan en esta parte deben ser consistentes con las definidas en el encabezado.

Epílogo: contiene instrucciones de procesamiento y comentarios al final del documento.

La figura 2.4 muestra un documento XML para el ejemplo de Correo Electrónico. A continuación se explican las principales características sintácticas del documento.

2.3 Elementos

Los elementos son los bloques con los que se construye la estructura jerárquica que conforma los datos, ya que los elementos pueden contener a otros elementos. Si se observa el ejemplo anterior,

[12] Diseñado como un superconjunto de la mayoría de los conjuntos de caracteres existente hasta el momento. Es congruente con el estándar ISO/IEC 10646

[13] Los modelos de contenido se explicarán en los capítulos 3 y 4.

el elemento Encabezado está compuesto por otros cuatro elementos: Destinatario, Remitente, Fecha y Asunto. Además, los elementos se delimitan con etiquetas que abren y cierran el elemento.

```
<?xml version= "1.0" standalone = "yes"?>
<! DOCTYPE Correo [
 <! ELEMENT Correo (Encabezado, Mensaje, Adjunto*)>
 <! ELEMENT Encabezado (Destinatario+, Remitente, Fecha,
           Asunto)>
 <! ELEMENT Mensaje (Linea+)>
 <! ELEMENT Adjunto (CDATA)>
    <! ATTLIST Adjunto tipo (#PCDATA)>
 <! ELEMENT Destinatario (#PCDATA)>
 <! ELEMENT Remitente (#PCDATA)>
 <! ELEMENT Fecha (#PCDATA)>
 <! ELEMENT Asunto (#PCDATA)>
 <! ELEMENT Linea (#PCDATA)>
    <! ATTLIST Linea numero (#PCDATA)>
]>
<Correo>
 <Encabezado>
   <Destinatario>xyz@abc.com</Destinatario>
   <Remitente>yoyo@servidor.com</Remitente>
   <Fecha>12-03-2002</Fecha>
   <Asunto>Ejemplo de Correo Electrónico</Asunto>
 </Encabezado>
 <Mensaje>
   <Linea numero="1">Estimado Colega:</Linea>
   <Linea numero="2">Le comunico que se está presentando:</Linea>
   <Linea numero="3">Un ejemplo de un documento XML</Linea>
   <Linea numero="4">Cualquier consulta me devuelve el mensaje</Linea>
 </Mensaje>
 <Adjunto tipo="txt">ejemplo.txt</Adjunto>
</Correo>
<?xml-stylesheet  href="miEstilo.css"  type="text/css"  title="Estilo"
media="screen"?>
```

Encabezado o prólogo

Cuerpo del documento

Epílogo

Figura 2.4 Ejemplo de documento XML para correo electrónico

2.4 Atributos

Algunas veces puede resultar de utilidad agregar información sobre los elementos, no sobre su contenido. A esta metainformación se le llama *atributos*. En el ejemplo del documento de correo electrónico, se tiene que Adjunto tiene un atributo **tipo** y Línea tiene un atributo **número**, que describen el elemento en sí y no su contenido.

2.5 Instrucciones de procesamiento

Las instrucciones de procesamiento son el mecanismo que provee XML para adjuntar procesamiento o aplicaciones a los datos ya que puede ser muy útil adjuntar a los datos la hoja de estilo que les dé presentación o la dirección de la aplicación requerida para procesar los datos.

Como pudo verse, la sintaxis de XML es muy regular, presenta documentos estructurados mediante secuencias de caracteres estándar. Dicha regularidad facilita el procesamiento automático de los documentos y permite validar estrictamente su contenido. Los mecanismos para realizar lo anterior se presentan en los siguientes dos capítulos.

3 Definición de tipos de documentos

Este capítulo describe brevemente los principales conceptos de una DTD (*Document Type Definition*), que es un mecanismo provisto por XML para modelar el contenido de un conjunto de documentos y permitir su validación. Además de permitir la validación de documentos, las DTDs tienen la ventaja adicional de que hacen posible el desarrollo de herramientas genéricas que no están encadenadas a un modelo específico de documentos, sino que pueden validar documentos de cualquier modelo siempre y cuando reciban como entrada la DTD adecuada.

Las DTDs no son el único mecanismo que tiene XML para modelar y validar documentos. En el capítulo siguiente se presentan los *Schemas* (esquemas), que son una forma alternativa y más poderosa de hacer lo mismo.

3.1 Documentos XML

Según la Especificación de XML, un documento es un objeto de datos que tiene una estructura lógica y una estructura física. Físicamente, el documento está compuesto de unidades llamadas **entidades**. Una entidad puede referirse a otras entidades, lo que provoca su inclusión en el documento. Esto permite dividir el documento físicamente en entidades (archivos), cada uno de los cuales contiene alguna unidad significativa del documento mismo: secciones, encabezados, etc. Por el lado lógico, un documento está compuesto por declaraciones, elementos, comentarios, referencias a caracteres e instrucciones de procesamiento. Todos estos componentes se señalan dentro del documento usando etiquetas o marcas explícitas [W3C 1998].

Hay dos conceptos muy importantes en XML, el de documento **bien formado** y el de documento **válido**.

Un documento está *bien formado* si cumple con las siguientes condiciones de la Especificación de XML:

- cuenta con un único elemento, llamado raíz, en su bloque principal que no aparece contenido dentro de ningún otro elemento
- para todos los demás elementos sus etiquetas de inicio y final están anidadas apropiadamente; esto es, cada elemento está completamente contenido dentro del elemento superior
- todas las entidades usadas dentro del documento están **bien formadas**; o sea, cada elemento está completamente contenido dentro de una misma entidad
- cumple con restricciones adicionales de la Especificación sobre parámetros y otros aspectos [W3C 1998].

La figura 3.1 ilustra algunos de los conceptos que deben cumplir los documentos bien formados.

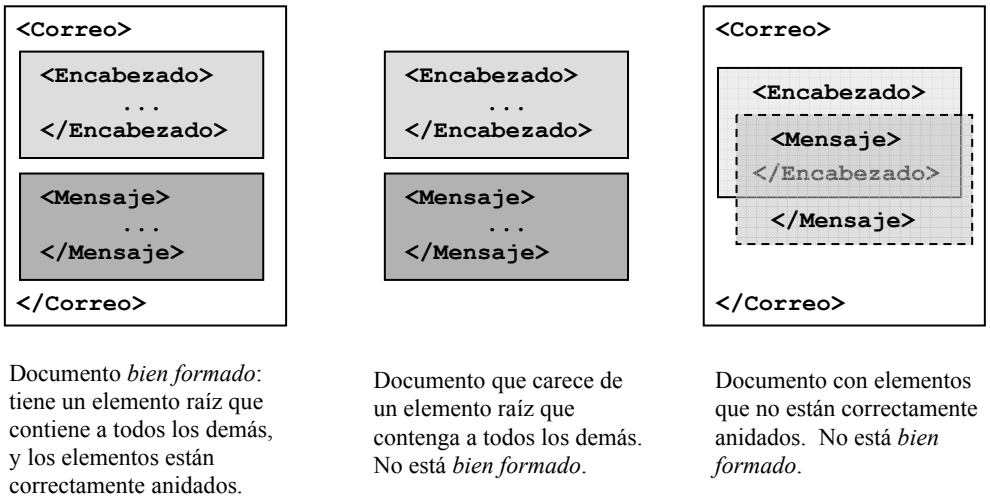


Figura 3.1 Algunos casos sobre documentos bien o mal formados

Un documento es **válido** si además de ser bien formado, cumple con las restricciones especificadas en la Definición de Tipo de Documento que tiene asociada ese documento. Las DTDs especifican cómo se relacionan los diferentes elementos; por ejemplo, qué elementos pueden aparecer dentro de un elemento específico, si pueden haber repeticiones de un elemento, el orden de aparición de los elementos dentro del elemento que los contiene, y cuáles elementos son opcionales.

3.2 Definición de tipos de documentos (DTD)

Para que el procesamiento de documentos sea confiable, es necesario describir clases de documentos y verificar la pertenencia de documentos individuales a estas clases. Como se mencionó al inicio de este capítulo, existen dos formas de definir la estructura de un documento XML; se puede hacer por medio de una DTD o por medio de un esquema. Una DTD es un conjunto de reglas que definen la estructura de un documento XML. Aunque XML hereda dicho mecanismo de SGML, las DTDs de XML son más simples que las de SGML y no contienen toda la funcionalidad de SGML. Esta simplificación elimina características de SGML que son difíciles de implementar y que no son importantes para el dominio de aplicación del XML.

La importancia de los DTD va más allá de la creación de documentos estructurados. Las DTDs ayudan a que diferentes personas y diferentes programas puedan leer los archivos de los demás. Por ejemplo, si un grupo de músicos acuerda una DTD sobre notación musical, entonces, la elaboración de documentos especializados en música que todos puedan comprender se vuelve más sencilla. Además, si esta estructura llegara a estandarizarse en la profesión, obligaría a los vendedores de software a mantenerse en el estándar abierto.

Dependiendo de su ubicación, una DTD puede ser interna, externa o mixta. Se dice que es interna, si reside dentro del encabezado del propio documento que modela. Por otro lado, se dice que es externa si está almacenada en un archivo separado; en este caso en el encabezado del documento debe haber alguna referencia que permita localizar al archivo que contiene la DTD. Finalmente, la DTD es mixta si una parte de ella se encuentra explícitamente dentro del documento pero además hace referencia a archivos que contienen el resto de la DTD. En este caso, cuando la declaración interna redefine una declaración, esta sustituye a la definición externa. El caso externo es el más común ya que permite la reutilización de las definiciones y todos los beneficios que esta conlleva.

Una DTD se define usando una declaración de tipos de documentos (`<!DOCTYPE ...>`) donde se definen el nombre del elemento raíz y las propiedades o atributos de los elementos que conforman el documento. Los elementos son definidos por medio de declaraciones `<!ELEMENT ... >`, las cuales permiten especificar el nombre de los elementos y su contenido. Un ejemplo sencillo de una DTD es el documento Hola Mundo que se presenta en la figura 3.2.

Documento Saludo con DTD Interna	Documento Saludo con DTD Externa	Documento Saludo con DTD Mixta

`<?xml version="1.0" standalone="yes"?>`
`<!DOCTYPE SALUDO [`
`<!ELEMENT SALUDO (#PCDATA)> ]>`
`<SALUDO>`
`Hola Mundo!`
`</SALUDO>`
	`<?xml version="1.0" standalone= "no"?>`
	`<!DOCTYPE SALUDO SYSTEM "ejemplo.dtd">`
	`<SALUDO>`
	`Hola Mundo!`
	`</SALUDO>`
		`<?xml version="1.0" standalone= "no"?>`
		`<!DOCTYPE SALUDO SYSTEM "ejemplo.dtd">`
		`<!DOCTYPE SALUDO`
		`<!ELEMENT SALUDO (#PCDATA)>`
		`]>`
		`<SALUDO>`
		`Hola Mundo!`
		`</SALUDO>`

Figura 3.2. Formas de especificar DTDs.

Estos ejemplos son muy sencillos y no muestran los alcances reales de las DTDs. En este documento no se profundizará más en el mecanismo de DTDs y solamente se resumen algunas de sus principales características en la Tabla 3.1. Si se desea profundizar más en el diseño de DTDs se puede encontrar mucha información en [Rusty 1999] y en [Martin 2000].

Para que un documento sea válido, su contenido debe satisfacer la estructura declarada en la DTD correspondiente. Esta validación la realiza un programa llamado "parser" o analizador léxico-sintáctico. Casi todas las herramientas XML incluyen internamente un parser. También se pueden encontrar analizadores como programas independientes en varios sitios Web y algunos incluso son gratuitos[14].

[14] Por ejemplo, XML para Java de IBM URL: http://www.alphaworks.ibm.com/tech/xml o XJParser de Microsoft y DataChannel, URL: http://www.datachannel.com/xml_resources/

Tabla 3.1. Resumen de las principales características de las DTDs
Una DTD provee la lista de elementos, etiquetas, atributos y entidades contenidas en un documento, y las relaciones entre ellos.
El prólogo de un documento puede contener una declaración de tipo de documentos que especifica el elemento raíz y la DTD.
En la DTD se establecen las etiquetas permitidas y la estructura del documento. Así, un documento que cumple con estas reglas se dice que es válido.
En las declaraciones de tipos de elementos se declaran el nombre y los hijos de un elemento. Los elementos pueden contener dos tipos de datos básicos: Simples, conocidos como #PCDATA (Parsed Character DATA que, en el fondo, es texto puro) y Compuestos (contienen otros elementos). No existen tipos de datos como los que se encuentran en los lenguajes de programación.
La declaración de elementos utiliza una gramática sencilla, por ejemplo, por medio de signos de puntuación se puede definir la propiedad de ocurrencia de un elemento, * significa que puede aparecer repetido cero o más veces, ? significa que puede o no aparecer y + significa que debe aparecer al menos una vez. Los elementos pueden contener listas o secuencias de otros elementos, lo que forma la jerarquía del documento. También se pueden definir con contenidos mixtos, es decir, otros elementos y texto. Al separar la lista de elementos hijos con comas se establece un orden estricto y con paréntesis se agrupan elementos hijos para detallar más la declaración.
En una DTD todo lo que no está explícitamente permitido está prohibido.
El orden en que aparecen las declaraciones es irrelevante siempre y cuando todas estén contenidas en la DTD.
Las referencias a DTDs externas se especifican con la palabra reservada SYSTEM.
Cuando se utilizan declaraciones de DTDs Externas e Internas, los conflictos de declaraciones se resuelven dando prioridad a la DTD Interna.

3.3 Limitaciones de las DTD

Las DTDs tienen limitaciones que han hecho necesario recurrir a otros recursos como son los Esquemas. Entre esas limitaciones está el hecho de que las DTDs siguen una sintaxis distinta de las sintaxis de los documentos. Las DTDs no se especifican en términos de etiquetas, ni elementos, ni atributos. Una limitación más seria de las DTDs es que son muy pobres en el manejo de tipos de datos. Esencialmente solo tienen el tipo texto y unos pocos tipos adicionales como número o identificador. Además, no permiten la definición de tipos compuestos basados en tipos más sencillos. Esta es una seria limitación puesto que uno de los objetivos de XML es permitir el intercambio de información entre aplicaciones. Para lograr esto, es fundamental que las aplicaciones puedan hacer una revisión adecuada de los tipos de la información que es intercambiada.

Los esquemas son presentados en el siguiente capítulo de este informe. Este mecanismo permite superar las limitaciones enumeradas anteriormente. Por un lado, los esquemas siguen la misma sintaxis de un documento XML. Por otro lado, los esquemas incluyen poderosos mecanismos de definición de tipos complejos. Finalmente, al contrario de las DTDs, es posible combinar varios esquemas independientes en uno solo. Esto facilita el manejo de información que contiene partes modeladas por diferentes agentes.

4 Esquemas y espacios de nombres

Las DTDs presentan serias limitaciones para representar las estructuras de datos requeridas por aplicaciones XML más sofisticadas. En particular, las DTDs son difíciles de escribir y entender porque utilizan un lenguaje distinto del lenguaje usado en los documentos. Por otro lado, las DTDs no disponen de un mecanismo de manejo de nombres que permita reutilizarlas fácilmente sin que se produzca un conflicto de nombres. Relacionado con esto, no es factible combinar múltiples DTDs para crear una estructura que contenga elementos cuyas descripciones provengan de diferentes estándares de la industria. Finalmente, con respecto de los tipos de datos, las DTDs proveen muy pocos tipos de datos y no hay disponible un mecanismo para derivar nuevos tipos de datos que reutilicen tipos de datos anteriores proveyéndoles de nueva funcionalidad.

Los *Esquemas* y los *Espacios de Nombres* son la solución que la W3C ha planteado para responder a las limitaciones anteriormente señaladas y satisfacer las necesidades de aplicaciones XML sofisticadas.

4.1 Esquemas

Los Esquemas XML (XML Schemas), son una forma alternativa para especificar la estructura particular de una clase de documentos. A diferencia de las DTDs, los esquemas utilizan la misma sintaxis de los documentos XML. De hecho, son una aplicación más de XML. Esto ofrece la ventaja de que no es necesario aprender otro lenguaje para definir los tipos de documentos. Los esquemas, además, permiten trabajar con tipos de datos más complejos y extensibles.

El esquema define los elementos que pueden aparecer dentro de un documento y los atributos que pueden asociarse a cada elemento. Define además la estructura de los documentos, incluyendo las relaciones jerárquicas entre elementos. También define los tipo de datos de cada elemento y la cardinalidad (frecuencia) con que los elementos pueden aparecer.

Así, han surgido diversas propuestas de lenguajes de esquemas basados en XML, entre ellos está XML-Data, DCD (Descripción de Contenidos de Documentos), SOX (Esquema para XML Orientado a Objetos) y DDML (Lenguaje de Etiquetas para Definiciones de Documentos). Estos estándares están documentados en el Anexo 2. Además, también el RDF (Estructura de Descripción de Recursos), surgió como una estrategia para describir recursos que puedan ser descubiertos automáticamente. En este documento se describirán los conceptos de esquemas según el estándar *XML Schemas*.

Un esquema está hecho de bloques llamados *componentes de esquemas*. Los componentes principales definen los tipos de datos que serán usados así como los elementos y los atributos que forman los documentos. Además de los componentes principales, hay otros componentes que permiten incluir anotaciones de documentación en el esquema y agrupar ciertas definiciones para facilitar la escritura del esquema.

En el Ejemplo 4.1 se muestra un esquema básico. En él se puede apreciar la diferencia de sintaxis con respecto de una DTD: el esquema es un documento XML. A diferencia de las DTDs, en los esquemas la frecuencia de repetición de un elemento se describe por medio de los atributos minOccurs y maxOccurs, lo cual permite un control más preciso que el provisto por los signos de puntuación de las DTDs (* + ?).

```
< Schema targetNS="http://servidor/correo.xsd"
         version="1.0"
         xmlns="http://www.w3.org/1999/XMLSchema">

<xsd:element name="correo">
   <xsd:ComplexType>
      <xsd:sequence>
         <xsd:element name="encabezado" type="EncabezadoType"/>
         <xsd:element ref="adjunto" minOcurrs="0"
                                    maxOcurrs="unbounded"/>
      </xsd:sequence>
   </xsd:complexType>
</xsd:element>

<xsd:complexType name="EncabezadoType">
   <xsd:sequence>
      <xsd:element name="destinatario" minOcurrs="1"
                                       maxOcurrs="unbounded" />
      <xsd:element name="remitente" type="xsd:string">
      <xsd:element name="prioridad">
         <xsd:simpleType>
            <xsd:restriction base="xsd:string">
               <xsd:enumeration value="normal"/>
               <xsd:enumeration value="urgente"/>
            </xsd:restriction>
         </xsd:simpleType>
      </xsd:element>
   </xsd:sequence>
</xsd:complexType>

<xsd:group name="adjunto" order="xsd:seq">
  <xsd:element name="nombreadj" type="xsd:string"/>
  <xsd:element name="numeroadj" type="xsd:integer"/>
</xsd:group>

<xsd:element name="linea"   type="xsd:string"/>

</Schema>
```

Ejemplo 4.1 Esquema básico

El Ejemplo 4.1 ilustra un esquema que define una estructura simplificada para el manejo de correo. El elemento principal, llamado correo, contiene un encabezado y cero o más adjuntos. El encabezado, por su parte, contiene información sobre uno o más destinatarios, sobre el remitente y sobre la prioridad del mensaje.

Un esquema consiste de un preámbulo y cero o más definiciones y declaraciones. Como se muestra en el ejemplo anterior, en el preámbulo se indican, al menos, tres datos muy importantes: **targetNS**, que es el URI[15] del esquema que se está creando; **version**, para especificar la versión del esquema; y **xmlns**, para especificar el espacio de nombres en el que se basa el esquema. Los espacios de nombres serán descritos en la siguiente sección.

La declaración de elementos se realiza por medio de la etiqueta `<element ...>`. En ella se establecen las propiedades del elemento. En particular, se indica si un elemento es simple (contiene un único valor), o si se trata de un elemento compuesto (contiene varios valores). Por su lado, los tipos de datos se definen por medio de las etiquetas `<complexType>` y `<simpleType>`. A los tipos de datos se les puede dar un nombre para poder usarlos en las definiciones de múltiples elementos o pueden incluirse explícitamente dentro de un elemento para declarar su tipo.

El Ejemplo 4.1 muestra el uso de los atributos `minOccurs` y `maxOccurs` para controlar las repeticiones de algunos elementos. Además, se muestra cómo se pueden definir tipos como `EncabezadoType`. Luego, se muestra cómo establecer restricciones adicionales sobre los valores de un elemento; en el caso del elemento `prioridad` se estableció que solo dos valores son válidos: "`normal`" y "`urgente`". Finalmente, con el elemento "adjunto" se muestra una de las características más importantes de los esquemas como es la abstracción y reutilización de tipos. Como se puede ver en la definición de `correo`, después del elemento `encabezado` se incluye otro elemento cuya estructura interna se halla definida en otra parte. Esto se señala por medio del atributo `ref`, el cual referencia a la definición de `adjunto` que está más abajo. En la Tabla 4.1 se resumen los principales conceptos alrededor de Esquemas. El lector puede encontrar información más detallada sobre XML Schemas en [Martin 2000] y [Standefer 2001].

Tabla 4.1. Resumen de las principales características de XML Schemas
Son la respuesta más aceptada de la W3C para resolver las limitaciones de las DTDs. Entre sus ventajas están que siguen la misma sintaxis XML, permiten el uso de nombres de etiquetas definidas en archivos externos (espacios de nombres) y proveen una fuerte tipificación de contenido.
En el preámbulos del XML Schema se definen las propiedades del Schema en sí y en la declaración de tipos y elementos se definen las estructuras del Schema.
Los tipos pueden ser simples o compuestos. Para los elementos de tipo simple, existen tipos de datos primitivos como *string, boolean, float, double, decimal, timeInstant, timeDuration, recurringInstant, binary y uri*. Para los tipos compuestos, definidos por el diseñador, primero se asocia un modelo de contenido y luego se definen los atributos del tipo (que pueden ser otros elementos).
Los grupos son la estructura de construcción de definiciones de modelos de contenidos y permiten establecer estructuras anidadas complejas de tipos de datos.
Permite la derivación de tipos por medio de un conjunto de reglas de derivación como *derivation* y

[15] Un URI (Uniform Resource Identifier) es un nombre único para un recurso en la Web. Es más general que el URL (Uniform Resource Locator) que ubica recursos en términos de protocolos de acceso y redes.

composition. *Derivation* permite derivar un nuevo tipo a partir de otro existente y agregar contenido adicional o aumentar las restricciones sobre el tipo. *Composition* permite combinar esquemas y espacios de nombres para construir instancias de documentos basadas en varios esquemas; esto se realiza con los atributos *import* e *inclusion* en el preámbulo del esquema.

4.2 Espacios de nombres

A medida que crecen las aplicaciones y más desarrolladores conocen XML, se pueden encontrar, principalmente en Internet, gran cantidad de vocabularios que podrían ser útiles, total o parcialmente, en el modelado de cierta aplicación. Por lo tanto, resulta necesario contar con un medio para compartir y acceder a vocabularios disponibles en la red. La Recomendación de la W3C para satisfacer esta necesidad es la definición de *Espacios de Nombres* (*Namespaces*) como una colección de nombres, identificados por una referencia URI, que pueden ser utilizados en documentos XML como tipos de elementos y nombres de atributos.

Existe una sintaxis predefinida para la creación de *Espacios de Nombres*. La utilización de los *Espacios de nombres* requiere un poco más de análisis, ya que lo que resulta interesante es poder acceder a varios espacios de nombres en un mismo documento XML. Esto se logra por medio de nombres calificados y de alias.

En el Ejemplo 4.1 se ejemplifica el uso de un espacio de nombres en los prefijos xsd: que anteceden a las etiquetas que pertenecen al XMLSchema.

En la Tabla 4.2 se resumen los principales conceptos relacionados con Espacios de Nombres.

Tabla 4.2. Resumen de características de espacios de nombres
Son la respuesta más aceptada de la W3C para resolver los conflictos de nombres que probablemente surjan al combinar documentos XML provenientes de diferentes fuentes.
Los espacios de nombres se declaran mediante el atributo `xmlns`, el cual indica el URI asociado al espacio de nombres. Este atributo también define un prefijo que permite identificar cuándo se usan los conceptos de ese Espacio de Nombres.
Si un atributo `xmlns` no tiene prefijo entonces se considera espacio de nombres por omisión para todos los elementos y sus hijos (pero no para los atributos de los elementos).

Para resumir, XML Schemas y Namespaces responden a los problemas que surgen al implementar aplicaciones ambiciosas de XML. En particular, los esquemas y los espacios de nombres permiten a los desarrolladores:

- Organizar mejor los vocabularios que rodean a problemas complejos.
- Proveer un medio para retener los tipos de datos cuando se convierte de XML y hacia él.

- Describir vocabularios con más precisión y flexibilidad de lo que permiten las DTDs.
- Leer reglas de vocabularios en XML, permitiendo el acceso a definiciones de vocabulario sin incrementar la complejidad del parser [Martin 2000].

El Ejemplo 4.2 muestra el uso de espacios de nombres para combinar información proveniente de diferentes fuentes sin que se produzcan conflictos de nombres. En el ejemplo, se introduce información estructurada dentro de un mensaje de correo (como los definidos en el Capítulo 2). Dentro del elemento `Mensaje` se introduce una `Noticia` y un `Comentario`. Estos elementos a su vez, tienen subelementos que pueden entrar en contradicción con subelementos de `Correo`. Tal es el caso de `Fecha` en `Noticia` que no corresponde a `Fecha` en `Encabezado`, o `Asunto` en `Comentario` que es distinto a `Asunto` en `Encabezado`. Con el fin de evitar esos conflictos y poder validar el documento, se especifican dos espacios de nombres. El primer espacio se define al especificar el elemento `Noticia`. El atributo `xmlns` de dicho elemento especifica un espacio de nombres asociado al URI `http://www.noticias.com/noticia`, y cuyos elementos se identificarán por medio del prefijo `no`. En el caso de Comentario, el atributo `xmlns` especifica `http://www.coment.com/comentario` como URI y `co` como prefijo.

```
<Correo>
  <Encabezado>
    <Destinatario>juan@abc.com</Destinatario>
    <Remitente>luisa@servidor.com</Remitente>
    <Fecha>2004-09-15</Fecha>
    <Asunto>Favor leer noticia y comentario</Asunto>
  </Encabezado>
  <Mensaje>
    <no:Noticia xmlns:no="http://www.noticias.com/noticia">
      <no:Fecha>13 de setiembre del 2004</no:Fecha>
      <no:Lugar>San Jose</no:Lugar>
    </no:Noticia>
    <co:Comentario xmlns:co="http://www.coment.com/comentario">
      <co:Asunto>Pelicula El Padrino<co:/Asunto>
      <co:Texto>Una excelente pelicula</co:Texto>
    </co:Comentario>
  </Mensaje>
</Correo>
```

Ejemplo 4.2. Uso de espacios de nombres para combinar información de diferentes fuentes sin que hayan conflictos de nombres

En el próximo capítulo se descubrirán las fortalezas de XML relacionadas con el enlace y referencia de documentos.

5 Transformación y enlace de documentos XML

En este capítulo se describen las características más importantes de los siguientes estándares XML:

- XSL: estándar que establece mecanismos (hojas de estilo) para formatear y transformar documentos XML
- XSLT: parte del XSL que especifica cómo transformar documentos XML
- XLink: estándar que define los diferentes mecanismos de enlace entre documentos XML
- XPath y Xpointer estándares para especificar posiciones particulares de un documento XML

5.1 XSL

Un aspecto de XML que ha atraído la atención de la industria de software es su capacidad para separar el formato con que se muestran los datos, de la estructura en que están organizados dichos datos. Esto se logra mediante mecanismos de transformación que permiten satisfacer la necesidad de tener múltiples formas de ver los mismos datos.

El concepto de separar el formato de los datos no es nuevo. Anteriormente, la ISO/IEC definió el Lenguaje de Especificación y Semántica de Estilos de Documentos (DSSSL). Este es un lenguaje diseñado para SGML con características avanzadas para formatear documentos para su publicación. DSSSL tiene el inconveniente de estar basado en el lenguaje de programación Scheme, el cual no es muy conocido. Otro inconveniente de dicho lenguaje es no cuenta con una capa declarativa que permita que diferentes editores de hojas de estilos puedan interoperar.

Con la proliferación de HTML, surgió también el lenguaje de las Hojas de Estilo en Cascada (CSS). Si bien su aplicación inicial es el HTML, este lenguaje también puede utilizarse para aplicar estilos a documentos XML. Sin embargo, el CSS no tiene el poder de transformar y generar estructuras, como tablas de contenidos, que se requieren para la publicación de documentos basados en XML.

Es así como surge el *Lenguaje de Estilos Extensible* (XSL), el cual desde el principio formó parte del proyecto XML. Este lenguaje combina el poder de DSSSL, la simplicidad de XML y el vocabulario establecido previamente por las Hojas de Estilo en Cascada.

La combinación de XML con XSL tiene las siguientes características:

- Publicación de documentos independiente del medio.
- Independencia de las implementaciones particulares de HTML.
- Entrega de la misma fuente a múltiples usuarios, quienes pueden visualizarla de diferentes maneras.
- Menor número de visitas al servidor.

El XSL comprende básicamente dos lenguajes: uno de transformación llamado XSLT y otro de traducción llamado XSLF. El lenguaje de transformación permite mover datos de una representación XML a otra. Esto lo convierte en un componente muy importante para aplicaciones de comercio electrónico e intercambio electrónico de datos y metadatos. Por otro lado, el lenguaje de traducción permite traducir documentos de una representación XML a cualquier otra forma de representación. Esto resulta muy útil cuando se requiere representar un documento XML en diferentes plataformas o modelos de despliegue como voz, Braille, visual, etc.

La combinación de XML y XSL es potencialmente más compleja y difícil de dominar que la incorporación de CSS en HTML. Por esto, es muy probable que el XSL se aplique principalmente en ambientes con altos niveles de automatización e independencia de medios, como periódicos, directorios de negocios, enciclopedias, catálogos comerciales, programas de televisión, etc.

Se espera que poco a poco el enfoque estandarizado de separar formato y datos llegue hasta los programas de procesadores de palabras de escritorio, y que estos comiencen a salvar los archivos como combinaciones de XML y XSL en lugar de formatos propietarios. Pero esto solo sucederá cuando los usuarios exijan que los productos que compran utilicen formatos abiertos y estandarizados que puedan interoperar entre ellos.

Los beneficios de separar los datos de sus presentaciones son abrumadores:

- permite la interoperabilidad completa de contenidos y estilos a través de aplicaciones y plataformas
- libera a los creadores de contenidos del control de los vendedores de las herramientas de producción
- permite a los usuarios escoger sus propias vistas de los contenidos
- facilita la construcción de herramientas poderosas para la manipulación de contenidos a gran escala
- permite la búsqueda por contenido sin la distracción del formato/presentación
- abre oportunidades para desarrolladores de software independientes
- permite la publicación internacional real en cualquier medio.

En el Ejemplo 5.1 se presenta una hoja de estilos XSL, un documento XML al que se le aplicarán los estilos y el documento HTML que se obtiene finalmente. En la siguiente sección se explicará en detalle el ejemplo.

```
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:template match="Correo">
    <html><head><title>Mensaje</title></head>
    <body>
      <xsl:apply-templates/>
    </body></html>
  </xsl:template>
  <xsl:template match="Encabezado">
    <p>Para: <xsl:value-of select="Destinatario"/></p>
    <p>De:   <xsl:value-of select="Remitente"/></p>
  </xsl:template>
</xsl:stylesheet>
```

Hoja de estilo XSL

```
<Correo>
  <Encabezado>
    <Destinatario>xyz@abc.com</Destinatario>
    <Remitente>yoyo@servidor.com</Remitente>
    <Fecha>12-03-2002</Fecha>
    <Asunto>Ejemplo de Correo Electrónico</Asunto>
  </Encabezado>
  <Mensaje>
    <Linea numero="1">¡Saludos estimado colega!</Linea>
  </Mensaje>
</Correo>
```

Documento XML al que se aplicará la hoja de estilo anterior

```
<html><head><title>Mensaje</title></head>
<body>
<p>Para: xyz@abc.com</p>
<p>De:   yoyo@servidor.com </p>
</body></html>
```

Documento HTML resultante de la transformación

Ejemplo 5.1 Documentos XML asociados a la transformación del documento de correo electrónico a código HTML

5.2 XSLT

XSLT es el lenguaje de transformación de documentos XML más popular. Fue diseñado para usarse como parte de XSL. Como se ilustró en el ejemplo anterior, la sintaxis y la semántica del lenguaje se basan en transformaciones expresadas como hojas de estilo que a su vez son documentos XML bien formados conforme al espacio de nombres específico para los elementos de XSLT.

De acuerdo con la especificación del lenguaje, las transformaciones que se expresan en XSLT describen las reglas para trasformar un documento fuente en un documento resultado. La transformación es alcanzada mediante un conjunto de plantillas. Una plantilla consta de dos partes: la primera es un patrón, que calza con algunos elementos del documento fuente; por

ejemplo, **<xsl:template match="Encabezado">**, señala que dicha plantilla debe aplicarse a los elementos "Encabezado" del documento fuente. La segunda parte de una plantilla es un constructor que puede contener elementos literales que se incluyen directamente en el documento resultante; ese es el caso de los elementos "<html>" y "<head>" en el ejemplo. El constructor también puede contener elementos del espacio de nombres de XSLT que son instrucciones para crear partes anidadas del documento resultante usando partes anidadas del documento fuente. Por ejemplo, dentro de la plantilla mencionada anteriormente se encuentra la especificación **<xsl:value-of select="Destinatario"/>**, la cual especifica que se debe insertar en el documento resultante el texto contenido en el elemento "Destinatario". La estructura del documento resultante puede ser completamente diferente de la estructura del documento original. Durante la construcción del documento resultante, los elementos del documento fuente pueden ser filtrados y reorganizados.

A continuación se explicará en detalle la hoja de estilo del Ejemplo 5.1: El elemento <xsl:stylesheet> establece que este documento XML es una hoja de estilo y especifica el espacio de nombres correspondientes a las hojas de estilo. Esta hoja de estilo contiene dos plantillas, una para elementos <Correo> (<xsl:template match="Correo">) y otra para elementos <Encabezado> (<xsl:template match="Encabezado">). El atributo match establece las condiciones que debe cumplir un elemento para que se pueda usar dicha plantilla. En el caso de la plantilla para "Correo" el constructor especifica una serie de especificaciones HTML tales como <html>, <head>, <title> y <body> que serán incluidas literalmente en el documento resultante. Dentro de la plantilla de "Correo" se encuentra además la especificación <xsl:apply-templates/> la cual especifica que se debe buscar y aplicar plantillas para los elementos anidados dentro del elemento <Correo>.

5.3 Enlace de documentos XML

Con el fin de enriquecer los documentos XML, se requiere contar con mecanismos que, de manera estructurada, establezcan vínculos entre partes internas de un documento y que también permitan vincular a recursos externos. En caso de que el recurso externo accesado fuera un documento XML, se plantea la posibilidad de acceder a porciones específicas de este y no al documento como un todo.

La W3C ha propuesto 6 estándares claves para resolver estas necesidades de enlace y consulta:

- InfoSet o Conjunto de Información XML (XML Information Set)
- XLink
- XPath
- XPointer
- Intercambio de Fragmentos XML
- Consulta de Documentos XML

El propósito del InfoSet es proveer un vocabulario común que describa los contenidos de un documento XML. Establece que existen quince tipos distintos de información en un documento XML bien formado. En [Martin 2000] se puede encontrar la descripción completa de estos tipos. Este estándar es importante porque todos los demás estándares de enlace y consulta se basan en

la estructura de documento definida en él.

Las características de enlace están definidas por un lenguaje llamado *Lenguaje de Enlaces Extensibles* (XLL, en inglés eXtensible Linking Language). Este lenguaje está formado por dos componentes, XLink y XPointer. XLink define cómo serán las ligas entre documentos y XPointer define cómo se pueden acceder partes individuales de un documento [Rusty 1999]. Por su parte, XML Fragment Interchange define ciertos mecanismos para la creación y transferencia de porciones de un documento XML sin pérdida de contexto[16]. Por último, para resolver el problema de consulta de documentos se han definido varios estándares como XML-QL y XQL, que no han llegado a ser tan populares ya que también se puede resolver con una combinación de XSLT y XPath. A continuación se describirán brevemente los estándares XLink, XPointer y XPath.

5.4 XLink

Como se mencionó antes, XLink provee un mecanismo flexible para la definición de enlaces en documentos XML. Este estándar permite crear elementos de enlace explícitos dentro del documento, o agregar atributos de enlace a un elemento "corriente" del documento. XLink se puede usar para crear enlaces como los de HTML o para ligar varios recursos de manera más compleja. También permite separar la información de los enlaces del contenido, lo que facilita la actualización independiente de los enlaces.

En la Tabla 5.1 se comparan las características de los enlaces HTML y los enlaces de XML. Se verá que los últimos están diseñados para aplicaciones más ambiciosas donde se requiera un mayor control del flujo entre documentos.

Tabla 5.1. Comparación de Enlaces de HTML y XML

HTML	XML
Apuntan a un solo documento	Puede apuntar a varios documentos
Para apuntar a partes específicas de un documento se requiere que previamente se incluyan "anclas" en dicho documento destino	Por medio del estándar **XPointer** permite el enlace a cualquier posición arbitraria de un documento XML, sin que tenga que señalarse explícitamente
No permite establecer relaciones o historias entre documentos, es decir, los enlaces son en una sola dirección	Soporta enlaces multidireccionales
Los enlaces se definen con elementos `<A>`	Cualquier elemento puede ser un enlace
	Se tienen enlaces **out-of-line,** donde los enlaces están almacenados en un documento separado

[16] La información de contexto se describe mediante los mecanismos IDREF e IDREFS.

En XML cualquier elemento puede ser un enlace; simplemente se le debe asociar el atributo `xlink:form`. Los enlaces pueden ser simples o extendidos. Los enlaces simples son muy semejantes a los enlaces de HTML, ver la parte (a) de la Figura 5.1. Por su lado, los enlaces extendidos van mucho más allá que los enlaces de HTML, ya que permiten ligas multidireccionales (locales o remotas) entre grupos de documentos, como se ilustra en la parte (b) de la Figura 5.1.

Los enlaces cuentan con ciertos atributos que permiten mantener información más detallada sobre el enlace. Algunos atributos son informativos propiamente como *content-title* y *content-role*, mientras que otros atributos, como *show, actuate y behavior*, describen la forma en que el recurso está asociado al enlace. Por ejemplo, el atributo *show* permite especificar si al recorrer el enlace se debe crear una nueva ventana, o si se debe reemplazar el contenido de la ventana original, o si se debe insertar el contenido del documento enlazado dentro del contenido actualmente visible.

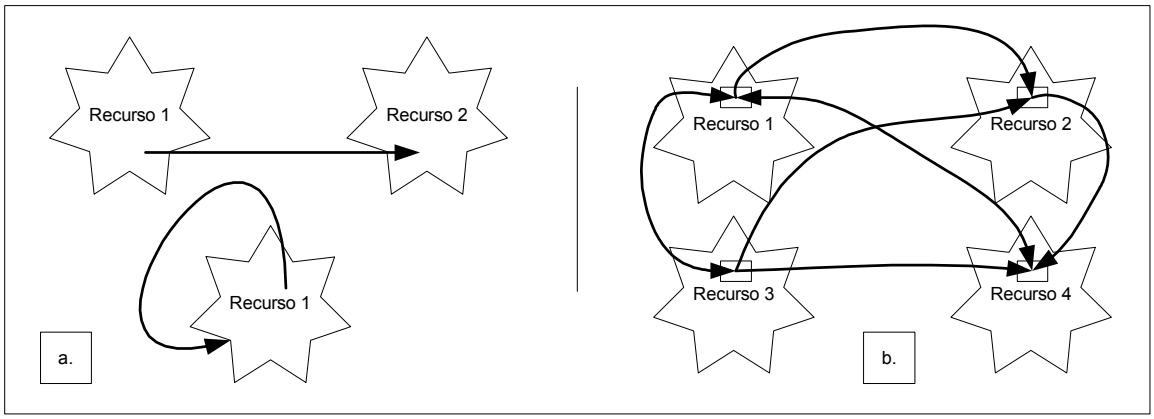


Figura 5.1 Esquemas de Enlaces Simples y Extendidos. *Adaptado de [Martin 2000]*

En el Ejemplo 5.2 se muestran un enlace simple en el elemento COMPOSER que apunta a un único destino, y un enlace extendido en el elemento WEBSITE que permite escoger entre un conjunto de destinos.

```
<COMPOSER xlink:form="simple"
      href="http://www.musicos.net/bandas.xml"
      role="lista de bandas" show="replace"
      actuate="onLoad">
   Lista de Bandas
</COMPOSER>
<WEBSITE xlink:form="extended"
         xlink:content-title="Sitios sobre Java"
         xlink:content-role="sitios de Java"
         xlink:title="Más sobre Java">
   <xlink:locator xlink:href="http://metalab.unc.edu/javafaq/"
    xlink:role="faq metalab" />
   <xlink:locator
    xlink:href="http://sunsite.univie.ac.at/jcca/mirrors/javafaq/"
      xlink:role="faq univie" />
   <xlink:locator
           xlink:href="http://sunsite.icm.edu.pl/java-corner/faq/"
      xlink:role="faq icm" />
   <xlink:arc xlink:from="faq metalab"
              xlink:to="faq univie"
              xlink:show="replace"
              xlink:actuate="onRequest"/>
   <xlink:arc xlink:from="faq univie"
              xlink:to="faq icm"
              xlink:actuate="onRequest"/>
</WEBSITE>
```

Ejemplo #5.2 Segmento de un Documento XML con enlaces.

En la Tabla 5.1 se resumen las características de XLink.

Tabla 5.1 Resumen de las características de XLink
Además de hacer todo lo que hacen los enlaces de HTML, proveen mecanismos adicionales como los enlaces a varias direcciones y los enlaces out-of-line.
Los enlaces se identifican con atributos *`xlink:form`* y *`href`*, con objetivos.
Otros atributos de los enlaces pueden indicar información complementaria del enlace o el comportamiento del enlace.
Los enlaces extendidos pueden incluir más de un URI y queda a la aplicación la decisión de cuál seleccionar entre las alternativas.
Los enlaces extendidos se pueden agrupar para conectar a un conjunto particular de documentos. Los enlaces pueden estar definidos en documentos externos y ser referenciados desde el documento principal mediante el rol `external-linkset`

Desde otro punto de vista, XLink es importante porque:

- Permite nuevas formas de asociación de información.
- Promueve la creación de estructuras de información avanzadas y de administración de sitios.
- Facilita la administración de documentos multimediales.

5.5 XPointer y XPath

El Lenguaje de apuntadores XPointer (XML Pointer Language, en inglés) es el mecanismo propuesto por la W3C para apuntar a una posición o porción particular de un documento sea esta un subárbol, un atributo o una secuencia de caracteres particular. También es el mecanismo para proporcionar vínculos a dicho contenido desde otros lugares y otros documentos. Específicamente, XPointer permite hacer referencia a un elemento específico de un documento XML en función de su contexto. XPath forma parte del estándar XPointer y es el método principal utilizado para la creación de enlaces y navegaciones en transformaciones XSLT. Así, XPath define un mecanismo de direccionamiento y XPointer provee una forma estándar de usar ese mecanismo en las referencias [Martin 2000]. Por lo tanto, XPointer provee una sintaxis para establecer la información de direccionamiento requerida en un enlace para apuntar a una posición específica de un documento XML.

La combinación de XLink con XPointer permite definir conexiones sofisticadas entre documentos. Se apunta a un documento usando Xlink, y con XPointer se selecciona un elemento particular dentro de ese documento. En el Ejemplo 5.3 se ejemplifican instancias de diferentes tipos de "xpointers" que se pueden especificar como parte de una URI. Como se puede ver, todas son referencias a diferentes posiciones en el mismo documento.

Dado que los documentos XML poseen estructuras jerárquicas anidadas, se pueden “navegar” de múltiples maneras y de ahí la definición de estos localizadores. XPointer está construido a partir de estos términos de localización que especifican el punto de referencia en el documento destino. Se puede utilizar como valor de localización el nombre de una etiqueta, un valor de contenido o un patrExisten términos de localización absolutos que no dependen del contenido del documento, como *id()*, *root()* y *html()*, Ejemplos 5.a y 5.b. También existen los términos de localización relativos a una posición dentro del documento, entre ellos *child*, *descendant*, *ancestor*, *preceding*, *following*, *psibling* y *fsibling*, como en los Ejemplos 5.c y 5.d. Los mecanismos anteriores se pueden combinar en localizadores compuestos ilustrados en el Ejemplo 5.e. El símbolo # previo al término indica que se debe devolver el documento a partir de ese punto, mientras que | indica que sólo debe devolver el elemento o los elementos solicitados.

```
a. http://www.w3.org/TR/1998/REC-xml-19980210.xml#root()
b. http://www.w3.org/TR/1998/REC-xml-19980210.xml|id(dt-xmldecl)
c. http://www.w3.org/TR/1998/REC-xml-19980210.xml#descendant(2, termref)
d. http://www.w3.org/TR/1998/REC-xml-19980210.xml#following(,termdef,
   term, CDATA Section)
e. http://www.zamo.com/genealogia.xml#root().child(6.PERSON).child(1,NAME)
```

Ejemplo 5.3 Segmento de un Documento XML con enlaces. *Tomado de [Martin 2000]*

El ejemplo anterior ha presentado solo un subconjunto de los mecanismos de XPointer, otros

mecanismos permiten realizar localizaciones más complejas. En [Rusty 1999] se presenta una descripción completa de XPointer.

XPath define un lenguaje de expresiones que permiten aislar determinados elementos de un documento. Estas expresiones utilizan cadenas de texto con formato especial, por ejemplo, valores asociados con atributos específicos (caso XSLT) o se agregan a identificadores uniformes de recursos (caso XPointer).

Una expresión completa de XPath se codifica como un trayecto de búsqueda, que se compone de uno o varios pasos de búsqueda delimitados, de la misma manera en que se delimitan las rutas de acceso en los sistemas de archivos, utilizando el "/". Cada paso de búsqueda en un trayecto de búsqueda es, en sí mismo, una expresión XPath compuesta de tres componentes: un eje, una comprobación de nodo y un predicado[17]. El eje indica al procesador en qué dirección mirar a partir del nodo contexto. La parte de comprobación limita la vista de nodos disponibles, ya sea por tipo de nodo, nombre o instrucción de procesamiento específica. El predicado es una expresión lógica que permite refinar el conjunto de nodos seleccionados por la expresión.

Algunos ejemplos de expresiones XPath son:

- `self::node()[puesto = "Editor"]`
- `//revision/attribute::fecha`
- `//revision/@fecha[. = "2004-11-11"]`
- `//empleado[descendant::supervisor/@identificacion = "102411"]`
- `//apellido[../following-sibbling::trab/proy_act/idproy="INFORME"]`

En algunos de los ejemplos se utilizan abreviaturas propias del lenguaje.

Además, XPath define un conjunto de funciones para la manipulación de contenidos. Estas funciones se pueden aplicar a conjuntos de nodos o a hileras de caracteres y pueden devolver como resultado conjuntos de nodos, hileras de caracteres, números o valores booleanos (falso o verdadero). Una referencia completa de las expresiones y funciones XPath la puede encontrar en [Williamson 2001].

En resumen, los estándares XML creados por la industria, tratan a los documentos XML de manera integral, ofreciendo mecanismos muy complejos y poderosos para diseñar, construir, transformar, enlazar y consultar documentos

[17] La sintaxis para este mecanismo es eje**::**comprobación_de_nodo[predicado]

6 Aplicaciones de XML

Este capítulo presenta al inicio los principales componentes de una aplicación XML. Luego se presentan con más detalle las consideraciones más relevantes de aplicaciones XML en las áreas de bases de datos, aplicaciones servidor a servidor, comercio electrónico y servicios Web. El capítulo finaliza con una presentación de los aspectos fundamentales de seguridad en XML: encriptación y firmas digitales.

6.1 Componentes e infraestructura de una aplicación XML

Considerando la aplicabilidad potencialmente amplia de XML, existen muchas concepciones sobre lo que es una aplicación XML. Así, XML se puede ver como Estructura de Datos (utilización tradicional de SGML, permite definir familias de aplicaciones); se puede utilizar en arquitectura de redes o aplicaciones cliente-servidor (como elemento de integración), también se puede ver como una aplicación de desarrollo de software tradicional (como componente de integración y comunicación).

Sin importar el dominio, toda aplicación XML requiere los siguientes procesos:

- Creación y Edición de esquemas o DTDs
- Validación y procesamiento en ambientes de ejecución
- Mapeo y transformación de documentos XML
- Almacenamiento, administración y distribución de conjuntos de datos XML

Así, se puede establecer una arquitectura básica de las aplicaciones XML que se muestra en la siguiente Figura 6.1. En esta arquitectura se reconoce básicamente un esquema de tres capas donde XML figura como intermediario entre la interfaz de usuario y la capa de lógica del negocio. Además, es en el lado del cliente donde se generan las vistas de los datos.

En cuanto a la transformación de XML a HTML, un procesador XSL se encarga de tomar los datos XML y aplicarle los formatos que están definidos en la hoja de estilo XSL, como se explicó en secciones anteriores el resultado puede ser HTML o un formato para otros medios. Este proceso se ilustra en la Figura 6.2.

Una aplicación XML simple puede consistir en la generación de páginas web a partir de documentos más o menos estáticos. En este caso, los documentos originales de la aplicación pueden estar directamente basados en XML, o pueden estar basados en algún formato distinto. En el primer caso, los documentos pueden ser procesados directamente por componentes que usen XML, mientras que en el segundo caso, se requiere de un componente de software, llamado generador por algunos sistemas, que se encarga de generar un documento XML. A partir de este punto, un procesador de hojas de estilo XSL puede generar múltiples productos: páginas HTML, documentos PDF o incluso páginas Voice XML que facilitan su lectura automática para personas con problemas de visión. La Figura 6.2 también ilustra estos conceptos.

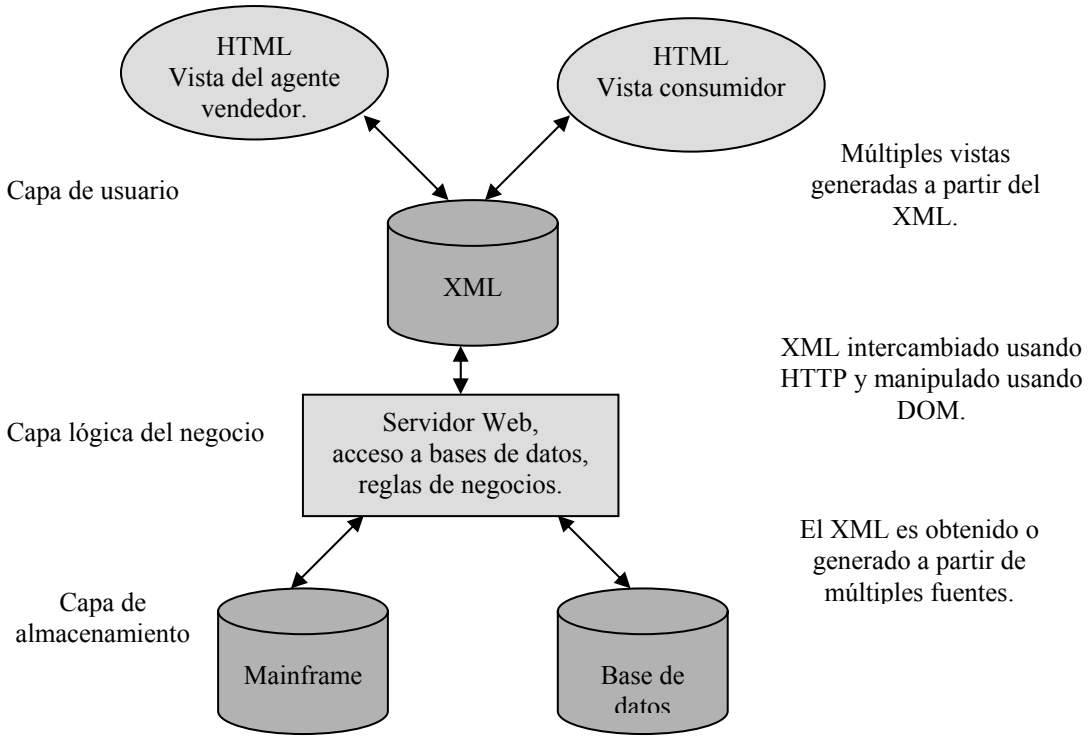


Figura 6.1 Arquitectura de XML.

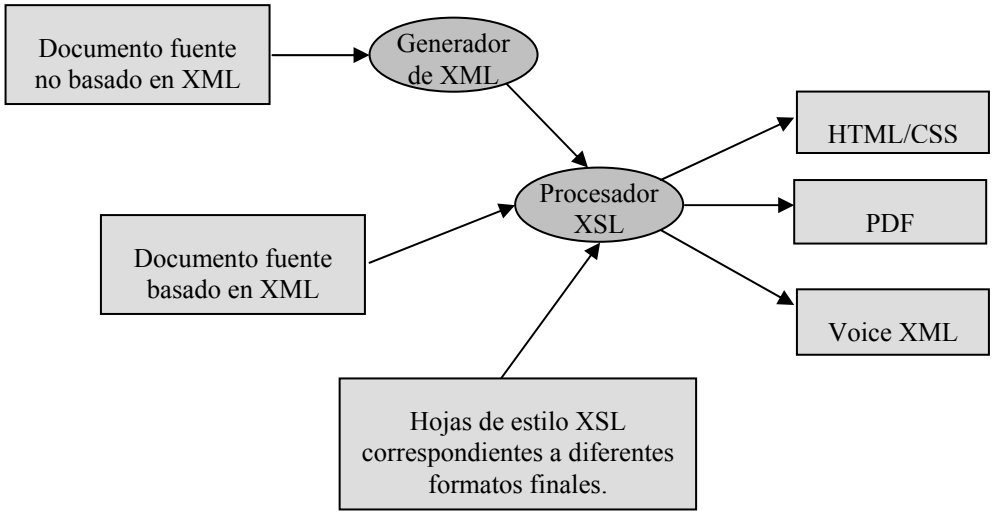


Figura 6.2. Aplicación simple de XML

Aplicaciones más complejas de XML pueden requerir de varias transformaciones que accedan a diferentes fuentes de datos tal como se muestra en la Figura 6.3 . En ese caso, se parte de un

documento XML que incluye dos aspectos: una consulta a una base de datos, y un conjunto de indicaciones para enviar la respuesta de la consulta por medio de correo electrónico. En primera instancia un transformador toma el documento y usa aquellas partes del documento que describen la consulta, para generar una consulta SQL que acceda a una base de datos y extraiga una respuesta. Dicha respuesta es convertida por ese transformador en parte de un documento XML que incluye además las indicaciones originales sobre el correo electrónico. Un transformador adicional se encarga de tomar este nuevo documento y producir el mensaje de correo [Langham 2003].

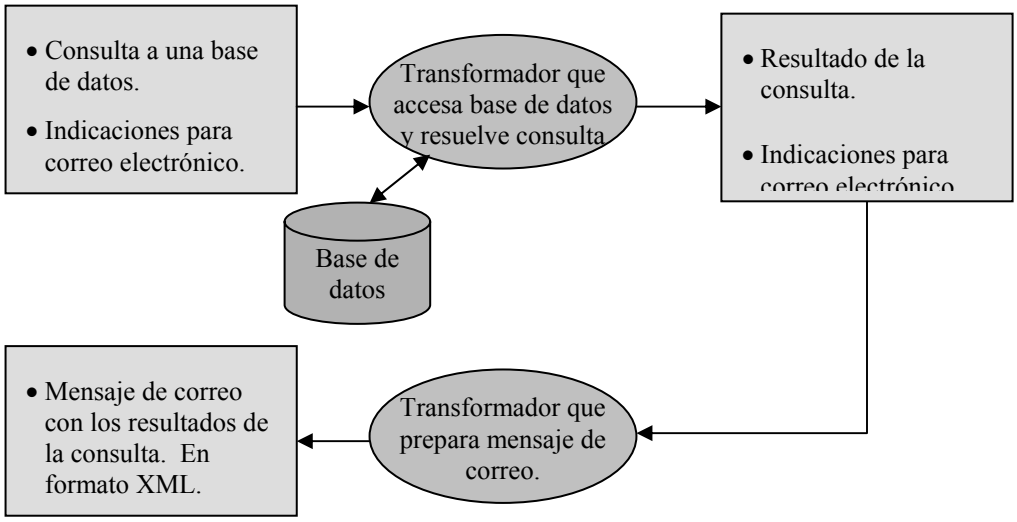


Figura 6.3. Aplicación de XML con varios transformadores

6.2 Bases de Datos

Las interfaces Web y las bases de datos relacionales son muy comunes. El modelo de datos de XML es inherentemente jerárquico y es difícil calzarlo con el modelo relacional utilizado por la mayoría de los administradores de bases de datos. Aunque las interfaces XML con datos relacionales no hacen uso directo de las características únicas de XML, es posible utilizarlo como mecanismo de integración independiente de la plataforma y de otras aplicaciones.

XML lo que ofrece es la capacidad de generar documentos XML automáticamente a partir de datos almacenados en diversos medios e intercambiar información entre diferentes almacenes de datos.

Este es un campo muy rico para la investigación, entre cuyas principales áreas está el almacenamiento de datos XML. Se deben considerar aspectos como tamaño de los documentos XML, concurrencia, la existencia de herramientas adecuadas para la administración de documentos XML, la administración de versiones, seguridad e integración.

Además, la existencia de diferentes sistemas administradores de bases de datos como relacionales, relacional-objeto y orientados a objetos requiere que las técnicas de almacenamiento y recuperación de datos XML sean adaptados a estos sistemas, o que los sistemas se adapten e incorporen XML.

Una de las aplicaciones más importantes de XML en bases de datos es el intercambio de información, donde XML desempeña un papel trascendental ya que provee un estándar para el intercambio y consulta de datos entre diferentes plataformas o ambientes de bases de datos. Además, XML le da un nuevo alcance a los sistemas cliente-servidor, porque diversifica el papel de los participantes y les permite ser clientes o servidores de otros participantes.

6.3 Servidor a servidor

Las aplicaciones Web basadas en XML pueden conectarse para construir sistemas. Hasta ahora, las aplicaciones WEB se veían estrictamente como aplicaciones Cliente-Servidor. La tendencia actual es de verlas como aplicaciones distribuidas con múltiples plataformas y sistemas. En estos ambientes XML ofrece el nivel de abstracción requerido para integrar los sistemas, concentrándose en la estructura de los datos.

Surgen entonces nuevos estándares para la manipulación de XML residente en servidores como:

- XML-RPC (XML Remote Procedure Calls),
- SOAP (Simple Object Access Protocol),
- WDDX (Web Distributed Data Exchange),
- SAX (Simple API for XML) y
- JAXP (Java for XML Protocol).

Por ejemplo, el XML-RPC permite que aplicaciones que corren en el servidor de una empresa invoquen por medio de mensajes XML, operaciones provistas por el servidor de sus proveedores. Por su lado, SAX y JAXP permiten desarrollar aplicaciones independientes de plataformas de los servidores.

6.4 Comercio electrónico

XML puede ser útil para definir el contenido de las aplicaciones Web, adaptación de datos de diferentes dispositivos clientes, integración de datos de diferentes fuentes, soporte de intercambio de datos automáticos entre diferentes aplicaciones, soporte de información personalizada.

Así, XML tiene una gran variedad de aplicaciones en Comercio Electrónico como:

- XML y Dispositivos Inalámbricos (M-Commerce);
- XML y Portales Empresariales;
- Integración de Negocios (Infraestructuras de Mensajería);
- Desarrollo de Aplicaciones (Formato de Intercambio de Metadatos, como XMI);
- Automatización de procesos (aplicaciones negocio-a-negocio), e
- Información Compartida (Cadena de Suplidores).

Los nuevos sistemas de EDI (Electronic Data Interchange) incorporan XML para interoperabilidad y agentes inteligentes (programas que actúan independientemente para alcanzar un objetivo para un usuario o para otro programa). Por ejemplo, localizar de entre un grupo de proveedores, aquel que ofrece el mejor precio por un producto.

6.5 Web Services

Los servicios Web son aplicaciones de XML que permiten describir, localizar e interactuar con programas, objetos o bases de datos ofrecidos por distintos agentes e integrarlos en una aplicación distribuida. Los servicios Web permiten abarcar sistemas tales como Sistemas adminstradores de bases de datos, .NET, J2EE, CORBA y otros, de modo que proyecten hacia la red una manera estándar de interactuar. Las interfaces de los servicios Web se encargan de recibir mensajes XML estándar, convertirlos a un formato comprendido por el sistema envuelto y devolver un resultado en otro mensaje XML si fuera del caso.

Hay dos tipos principales de interacción provistos por los Servicios Web [Newcomer 2002]:

1. **Llamado a procedimiento remoto**, RPC (procesamiento en línea): corresponde a invocar una función específica de una aplicación o un base de datos, especificando un conjunto de parámetros de entrada y de salida; por ejemplo, una consulta sobre la disponibilidad de un ítem; el agente que hace la consulta espera por el resultado. Este mecanismo se ilustra en la Figura 6.4.

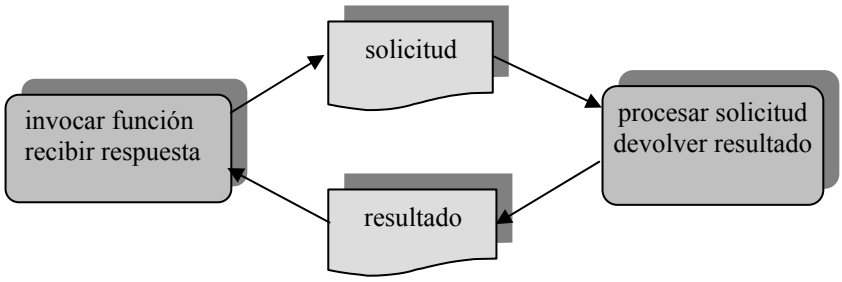
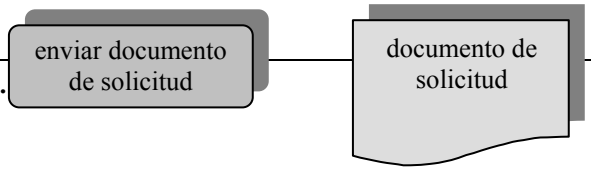


Figura 6.4. Interacción tipo RPC.

2. **Orientado al intercambio de documentos** (procesamiento en lote): corresponde a una solicitud que toma la forma de un documento XML completo que se espera sea procesado en forma completa; por ejemplo, el envío de una orden de compra completa que incluya varios ítemes; el agente que hace la solicitud solo espera a que la orden sea incluida en una cola de procesamiento y que siga el flujo predefinido para esos casos. Usualmente este método supone que las partes han alcanzado un acuerdo para compartir un mismo documento: orden de compra, factura, etc. Este mecanismo se ilustra en la Figura 6.5.



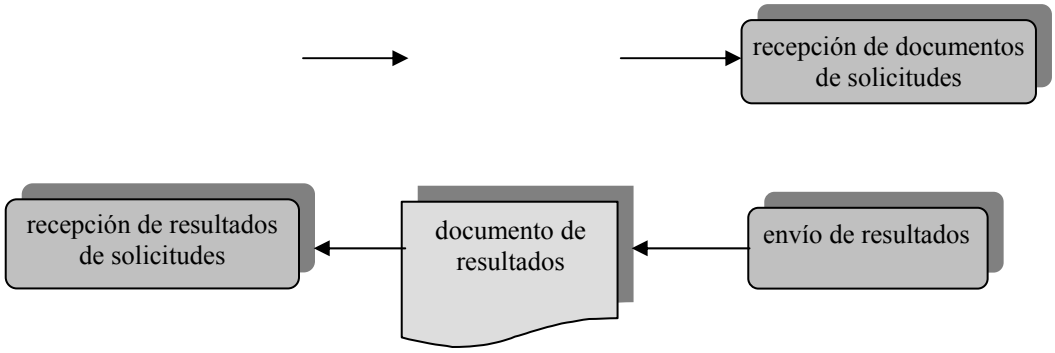


Figura 6.5 Interacción tipo intercambio de documentos

El establecimiento de servicios Web requiere de las tecnologías que se listan a continuación:

- **XML (Extensible Markup Language)**: provee el lenguaje para definir los datos intercambiados y cómo procesarlos
- **WSDL (Web Services Description Language)**: define, por medio de XML, las interfaces de los servicios, los tipos de los datos y de los mensajes, y los patrones de interacción empleados
- **SOAP (Simple Object Access Protocol)**: define, por medio de XML, una envoltura de comunicación para los Servicios Web que permita usar protocolos de transporte como HTTP; provee además de un formato de serialización para la transmisión de documentos y de una convención para describir interacciones tipo RPC.
- **UDDI (Universal Description, Discovery, and Integration)**: un registro de Servicios Web junto con un mecanismo de localización y búsqueda; almacena y categoriza información de negocios por medio de XML, y permite obtener referencias a interfaces de Servicios Web.

La Figura 6.6 ilustra el proceso de comunicación que tiene lugar al invocar solicitudes a un *Servicio Web*.

En las siguientes secciones se describen con más detalles las diferentes tecnologías involucradas en los Servicios Web.

El sistema que envía convierte su solicitud en un documento XML que invoca a alguna operación de un servicio. El documento contiene datos que cumplen con las especificaciones que describen el servicio y sus operaciones. WSDL describe el servicio y sus operaciones; SOAP convierte las solicitudes en documentos XML.

El sistema que recibe conoce por medio de WSDL y SOAP cómo interpretar el mensaje transmitido en el documento XML y así extraer la solicitud realizada. Para satisfacer la solicitud en forma concreta se debe ejecutar alguna aplicación, o se debe acceder alguna base datos.

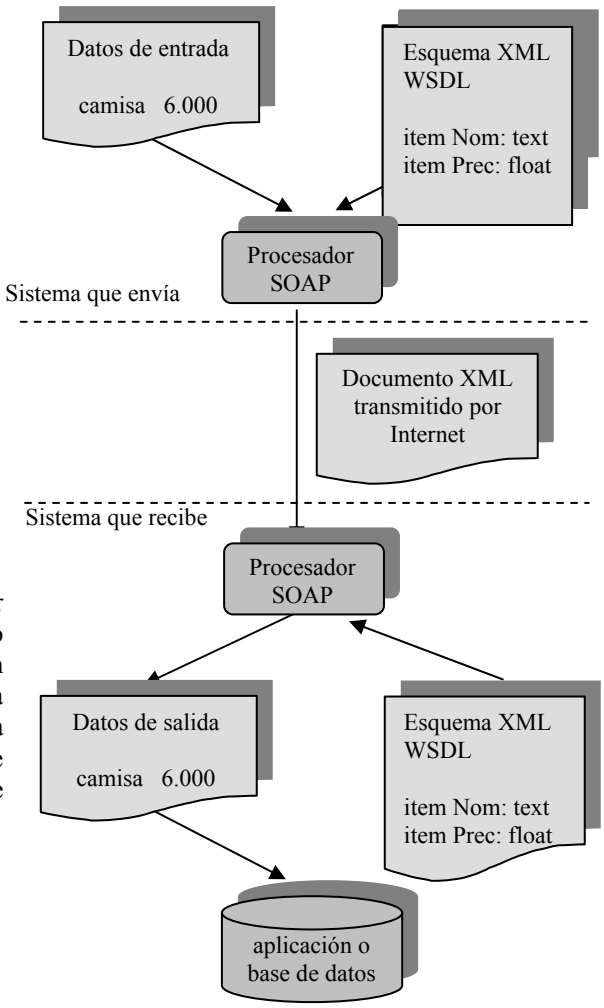


Figura 6.6 Proceso de comunicación en un servicio Web

6.5.1 WSDL (Web Services Description Language)

El WSDL consiste de un esquema de XML que provee un marco para describir las interfaces de los servicios Web. Dicho marco puede ser ampliado por los usuarios. Por medio del WSDL se describen los siguientes aspectos de los Servicios Web:

- los tipos de datos (estructura y contenido) de los ítemes de información pasados en los mensajes
- las operaciones que se pide aplicar al contenido de los mensajes
- las ligas que especifican qué capa de transporte de red acarreará a los mensajes.

Cada una de estas tres partes puede ser un documento XML independiente, o bien se pueden agrupar todas en un único documento XML.

A continuación se muestra la estructura lógica de un servicio llamado ServicioOrdenCompra

Se definen los tipos de datos que serán usados para intercambiar información:

OrdenCompra e ***Item*** son dos tipos de datos que contienen campos que almacenan la información que se desea intercambiar.

```
<schema  targetNameSpace="ServicioOrdenCompra-
xsd" >
   Por medio de un esquema se definen los
   tipos de datos que describen la información
   que será transferida por los mensajes.
   Por ejemplo:
      Item: contiene Precio, IDParte,
            Descripción y Cantidad
      OrdenCompra:
            contiene Compañía, Item y
            Dirección
</schema>
```

Se definen los contenidos de los mensajes que serán usados para intercambiar información:

SolicitudOrdenCompra
ResultadoOrdenCompra

```
<message name="SolicitudOrdenCompra">
   Envía una OrdenCompra
</message>
```

```
<message name=" ResultadoOrdenCompra">
   Recibe un valor que indica el resultado de
   la solicitud.
</message>
```

Describe operaciones indicando sus mensajes de entrada y salida.
Agrupa operaciones en puertos que luego serán asociados con algún mecanismo de transmisión.

Puerto: ***PuertoOrdenCompra***
Operación: ***postOrdenCompra***

```
<portType name="PuertoOrdenCompra">

   Describe un conjunto de operaciones

   Por ejemplo
      La operación postOrdenCompra tiene una
      SolicitudOrdenCompra como mensaje de
      entrada, y un ResultadoOrdenCompra como
      mensaje de salida.

</portType>
```

Los bindings o ligas asocian un puerto (conjunto de operaciones) con mecanismos que permitan intercambiar mensajes para realizar las operaciones:

bindingOrdenCompra indica cómo será la comunicación para el puerto ***PuertoOrdenCompra***

```
<binding name="bindingOrdenCompra" ... >

   Especifica el estilo de comunicacion y el
   protocolo de transporte usado para
   transmitir un conjunto de operaciones.

   Para cada operación describe aspectos como
   su codificación.

   Por ejemplo
      Se usará el protocolo http y el modo de
      interación RPC con el conjunto de
      operaciones PuertoOrdenCompra.

</binding>
```

6.5.2 SOAP (Simple Object Access Protocol)

Este estándar provee un marco para el intercambio en Internet de mensajes con formato XML. Entre sus objetivos está el ser simple y neutral en cuanto a la plataforma de sistema operativo, lenguaje de programación o plataforma de computación distribuida. Es como una extensión al protocolo HTTP para transportar mensajes XML. Para manejar los mensajes XML en forma correcta, los procesadores de HTTP deben disponer de la capacidad de manejar la especificación SOAP.

Las interacciones SOAP ocurren no solo entre un nodo que emite un mensaje y un nodo que lo recibe, sino que nodos intermedios pueden manejar casos especiales cuando el mensaje incluye encabezados particulares.

Las partes principales de un mensaje SOAP son:

- **Sobre** (*envelope*): define el inicio y el fin del mensaje SOAP.
- **Encabezado** (*header*):contiene atributos opcionales usados para procesar el mensaje ya sea por algún intermediario o en el punto final.
- **Cuerpo** (*body*): contiene los datos XML que forman el mensaje enviado.

A continuación se señalan las partes anteriores para un caso concreto de un mensaje SOAP tomado de [Newcomer 2002]:

Los componentes del mensaje, toman su definición
El encabezado **`Header`** especifica el espacio de nombres del *Web Service* a usar, el cual permite especificar una lista de medios de comunicación.

El cuerpo **`Body`** hace referencia a alguna operación definida por el *Web Service* anterior y especifica el contenido de la información pasada a dicha operación. En este caso el servicio consiste en enviar el mensaje dado a alguno de los medios especificados antes.

```
<env:Envelope
   xmlns:env="http://www.w3.org/2001/12/soap-envelope">
 <env:Header>
   <n:broadcastService xmlns:n=
          "http://www.xmlbus.com/broadcastServices">
     <n:list>PDA, Cell, Email, VoiceMail,IM
     </n:list>
   </n:broadcastService>
 </env:Header>

 <env:Body>
   <m:Function
      xmlns:m=
         "http://xmlbus.com/broadcastServices/send">
     <m:message>
       Eric, estas retrasado de nuevo!
     </m:message>
   </m:Function>
  </env:Body>

</env:Envelope>
```

6.5.3 Universal Description Discovery and Integration (UDDI)

Es una guía de servicios que permite localizar y usar servicios Web. Un servicio UDDI provisto por alguna compañía consiste de una base de datos, accesible públicamente, que contiene información sobre negocios y servicios. El servicio UDDI empezó como iniciativa de IBM,

Microsoft y otras compañías, quienes fundaron UDDI.org con el fin de establecer una guía de servicios web.

Para que un sevicio web sea incluido en el UDDI, se debe registrar la información con alguna de las compañías que proveen la base de datos de registro (IBM, Microsoft, HP, SAP). La información incluida en una de estas bases de datos es eventualmente replicada en las otras de modo que es posible localizar el nuevo servicio consultando a cualquiera de las bases de datos de registro. Por razones de seguridad, la actualización de los datos requiere contactar de nuevo al punto usado originalmente para introducir los datos. Antes de que un servicio pueda registrarse, se debe pasar por una etapa de aprobación.

Las dos funciones principales del UDDI son el registro y la localización de servicios Web. Para llevarlas a cabo, los operadores que proveen servicios UDDI usan WSDL separadas para describir esos dos servicios.

UDDI contiene tres categorías principales de información:

- **Páginas blancas**: nombre y dirección del negocio, información del contacto, sitio Web y cualquier otra información de identificación.
- **Páginas amarillas**: categorización taxonómica del tipo de negocio, de su localización geográfica y de sus productos.
- **Páginas verdes**: información técnica sobre cómo interactuar con los servicios provistos por un negocio.

La Figura 6.7 describe el modelo de datos básico usado por UDDI.

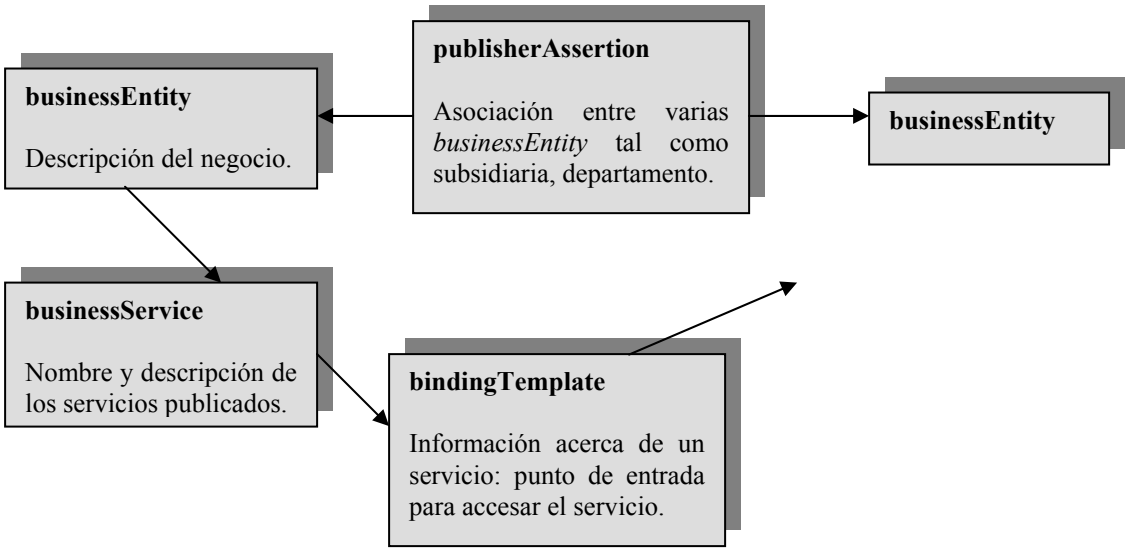


Figura 6.7. Modelo de datos básico de UDDI.

Por ejemplo, si una compañía de bienes raíces, llamarla XY, quisiera aprovechar este mecanismo, debe hacer lo siguiente:

- contactar a un proveedor de servicio UDDI (IBM, Microsoft, etc.)

- registrar la información del negocio correspondientes a la página blanca: nombre, dirección, teléfono, contactos
- registrar información que describa el área de negocios; aquí se aprovechan las categorizaciones predefinidas por la industria: localización geográfica, tipo de propiedades que se transan y los servicios que ofrece el negocio; por ejemplo, servicio de compra y venta de fincas
- registrar la información técnica que permita utilizar el servicio de compra, y el servicio de venta.

6.6 Seguridad en XML

Para enviar datos XML en forma segura por la Internet se deben usar dos mecanismos fundamentales: encriptación[18] para mantener información confidencial fuera del alcance de terceros, y firmas digitales para garantizar autenticidad, integridad, compromiso y no repudio. La W3C ha definido un estándar de encriptación para representar, usando XML, recursos del Web encriptados usando algoritmos arbitrarios. Por otro lado, en un esfuerzo conjunto, la W3C y la IETF han desarrollado estándares para firmas digitales en XML. Un punto importante de estos estándares es que permiten firmar partes específicas del documento XML y dejan la posibilidad de modificar otras partes del documento. Por ejemplo, un proveedor puede enviar una cotización a un cliente, la cual contiene una parte firmada por el proveedor y otra que el cliente debe llenar y firmar.

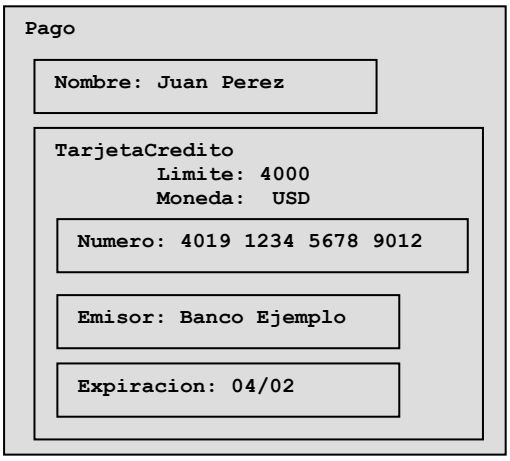
A continuación se presentarán más detalles sobre los estándares de encriptación y firma digital para XML.

6.6.1 Encriptación

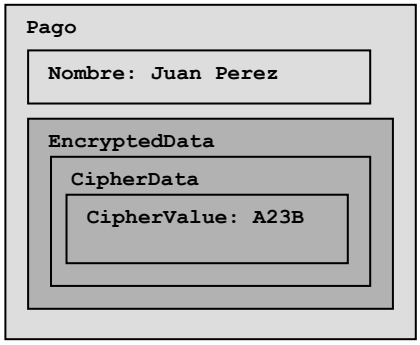
La especificación *XML Encryption Syntax and Processing* [W3C 2002] define el proceso para encriptar datos y representar el resultado en XML. Este estándar permite encriptar datos arbitrarios: documentos XML completos, elementos XML particulares dentro de un documento, o el contenido de un elemento. El producto de la encriptación es un elemento XML llamado `EncryptedData` el cual puede contener directamente los datos encriptados o por medio de un URI identificar su localización. Dicho elemento reemplaza al elemento o contenido que se quiere encriptar.

La Figura 6.8 ilustra un caso en que un pago incluye información confidencial sobre la tarjeta de crédito de un cliente. La parte (a) muestra la información completa sin encriptación alguna. La parte (b) ilustra el caso en que se encripta todo el elemento que representa a la tarjeta de crédito; se debe notar que se encripta todo el elemento `TarjetaCredito` de modo que no se puede accesar ni siquiera el nombre de dicho elemento. La parte (c) muestra la capacidad de encriptar el contenido de un elemento y no el elemento completo; en este caso, sí es visible el hecho de que hay un elemento llamado `TarjetaCredito` con atributos `Limite` y `Moneda`, pero el contenido de dicho elemento (`Numero`, `Emisor` y `Expiracion`) no es visible. Finalmente, en (d) se ilustra el caso en el que se encripta un elemento más restringido (`Numero`), y se hacen visibles los otros subelementos de `TarjetaCredito`.

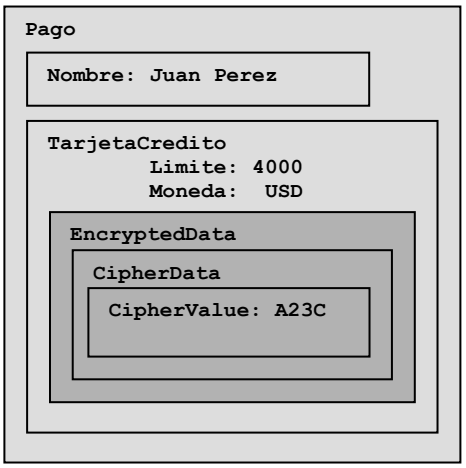
[18] Usaremos *encriptar* en lugar de *cifrar* por ser más común en Latinoamérica.



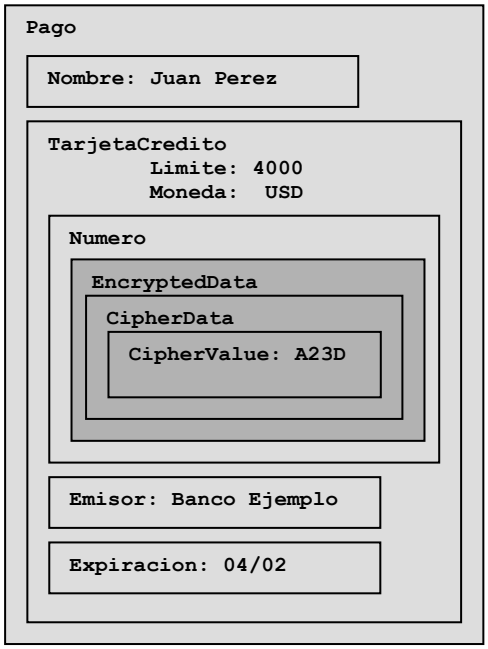
a) Documento sin encriptar.



b) Documento con el elemento **`TarjetaCredito`** totalmente encriptado. Ni siquiera se muestra el nombre del elemento encriptado.



c) Documento con el contenido del elemento **`TarjetaCredito`** encriptado. Se muestra el nombre del elemento, pero su contenido está encriptado



d) Documento con el contenido del elemento **`TarjetaCredito`** parcialmente encriptado

Figura 6.8 Encriptación de documentos XML.

6.6.2 Firmas digitales

Una firma digital es una sumarización (*hash*) encriptada de un documento que permite garantizar su integridad y autenticidad. También permite garantizar que la parte que firma no pueda repudiar su compromiso. La especificación *XML-Signature Syntax and Processing* [W3C 2001] define las reglas para crear y procesar firmas digitales en XML. Las firmas se pueden aplicar a datos dentro del mismo documento XML donde está la firma, o pueden referirse a datos en documentos externos. Una firma XML contiene la sumarización del documento y una descripción de los algoritmos usados para obtenerla. A continuación el Ejemplo 6.1 muestra un

caso de firmas digitales usadas en documentos XML. En la Tabla 6.1 se explican los principales conceptos asociados al caso. En este caso, la firma del ejemplo es externa, esto es, no se firma un elemento contenido en ese mismo documento sino que se firma el objeto que se referencia por medio del elemento <**Reference**>.

```
<Signature Id="MyFirstSignature"
           xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod
        Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod
        Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.w3.org/TR/2000/REC-xhtml1-20000126/">
      <Transforms>
        <Transform
          Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod
         Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>j6lwx3rvEPO0vKtMup4NbeVu8nk=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

Ejemplo 6.1 Documento XML para una firma digital

Tabla 6.1 Conceptos usados en el estándar de seguridad

Elemento	Explicación
`SignedInfo`	Contiene la información que es firmada.
`CanonicalizationMethod`	Algoritmo que transforma el elemento firmado a una forma canónica (eliminando espacios en blanco superfluos por ejemplo) antes de calcular la firma.
`SignatureMethod`	Algoritmo usado para convertir la forma canónica en el valor de la firma: `SignatureValue`. Es la combinación de diferentes algoritmos.
`Reference`	Identifica por medio de `URI`s cada objeto que es firmado; indica además cómo fue procesado cada objeto para obtener su firma. Los elementos `Transform` describen una secuencia de algoritmos que se deben aplicar antes de usar `DigestMethod` para calcular el valor asociado al objeto firmado.
`KeyInfo`	Indica la llave que se debe usar para validar la firma.

6.6.3 Otros mecanismos de seguridad

Los filtros inteligentes de contenido XML (*XML firewalls*) permiten un control de más alto nivel en cuanto al contenido de los mensajes que fluyen entre Internet y una red particular. Este control se realizaría por medio de transformaciones XSLT generadas dinámicamente a partir de reglas de más alto nivel [Callahan 2002]. Como se sugiere en [Kuznetsov 2002] los mensajes entrantes y salientes son examinados y varias pruebas de seguridad pueden ser aplicadas a ellos, tales como:

- revisar que el XML esté bien formado y que no hayan documentos inválidos que puedan afectar los sistemas
- proteger de ataques tipo denegación de servicios
- validar documentos usando estándares como SOAP o Esquemas o DTDs particulares
- verificar firmas
- encriptar partes sensibles de los documentos.

Para finalizar esta sección, las principales recomendaciones de seguridad para XML de acuerdo con [Salz 2003] son:

- Transformar todos los mensajes: aislar sistemas y crear dominios de seguridad internos y externos.
- Firmar todos los mensajes: proveer autenticación y prevenir la adulteración de mensajes.
- Validar todos los mensajes: validar mensajes entrantes y salientes para prevenir interrupciones en servidores.
- Protegerse de ataques tipo denegación de servicio: detectar XML que esté malformado y sea malicioso antes de que alcance los sistemas neurales (“back-end”).
- Encriptar los campos de los mensajes: proteger por encriptación sin limitar los procesos de negocios.
- Enmascarar los recursos internos: esconder URLs, direcciones IP, etc.
- Implementar filtrado XML: basado en el tamaño del mensaje, su contenido o información de la red.
- Bitácoras seguras: usar pistas de auditoría
- Poner etiquetas de tiempo en todos los mensajes.
- Usar capas de transporte seguras (SSL).

7 Conclusiones y Recomendaciones

- Aunque XML por sí mismo es reciente, está basado en una tecnología que tiene más de 30 años de investigación, por lo que sus bases teóricas son muy sólidas y amplias.

- El espectro de aplicaciones de XML es muy variado y se puede incorporar en cualquier área de conocimiento.

- XML propicia la proliferación de lenguajes de etiquetas de propósito específico que responden a las necesidades de un sector determinado.

- Aunque el lenguaje es sencillo, las tecnologías que tiene asociadas lo hacen más complejo y requiere un tiempo considerable la incorporación de estos conceptos en el ambiente laboral.

Debido a su importancia, es bueno repetir las ventajas ofrecidas por la separación de los datos y sus presentaciones:

- permite la interoperabilidad completa de contenidos y estilos a través de aplicaciones y plataformas
- da más control a los creadores de contenido
- permite a los usuarios escoger sus propias vistas de los contenidos
- facilita la construcción de herramientas poderosas para la manipulación de contenidos a gran escala
- permite la búsqueda por contenido sin las distracciones causadas por el formato o la presentación
- abre oportunidades para desarrolladores de software independientes
- permite la publicación internacional real en cualquier medio

XML permite obtener beneficios como:

- búsquedas más significativas;
- desarrollo de aplicaciones Web más flexibles;
- integración de datos de fuentes heterogéneas;
- intercambio de datos entre aplicaciones;
- datos de múltiples aplicaciones;
- cálculos y manipulación de datos localmente;
- variedad de vistas de los datos;
- actualizaciones granulares;
- estándares abiertos;
- formato para el despliegue en Web, lo que incrementa la escalabilidad y facilita la comprensión.

Para mantener el éxito de XML el W3C debe vigilar que:

- No se den estrategias que limiten XML al rol de middleware;
- Estar atento a los intentos de los vendedores de plataformas de dictar esquemas estándares y nombres de espacios;
- Estar atento a los intentos de los vendedores de navegadores de introducir extensiones fuera de los estandares a la familia de lenguajes XML;
- Estar atento a los vendedores de navegadores que introduzcan etiquedas ad-hoc en HTML bajo el nombre de XML;
- Dar soporte a vendedores de herramientas independientes de plataformas;
- Asegurar la interoperabilidad tanto de contenido como de estilo.

XML favorece la

- Neutralidad del vendedor
- Neutralidad de la plataforma
- Neutralidad del lenguaje.

XML ofrece importantes oportunidades a los desarrolladores de software independientes y en este momento es indispensable que en nuestra región se favorezcan las iniciativas de formación de nuevas empresas tecnológicas que implementen aplicaciones XML.

Es probable que XML y las tecnologías asociadas impacten más a los negocios en:

- independizar datos (contenido) de la presentación
- intercambio de datos entre aplicaciones
- integración de aplicaciones heterogéneas
- integración de apliaciones o intercambio de datos entre compañías
- participación como proveedores o consumidores de Servicios Web
- adopción de estándares que favorezcan el comercio digital seguro

El desarrollo de aplicaciones Web es una tendencia importante en nuestro país. Iniciativas gubernamentales como Impulso y Agenda Digital, exigen que las soluciones tecnológicas que se implementen en nuestras instituciones públicas incorporen mecanismos de comercio electrónico formalmente y fortalezcan las relaciones gobierno a gobierno, gobierno a público, gobierno con empresas, etc. Dada la diversidad tecnológica presente en nuestras oficinas gubernamentales, XML es la respuesta ideal de integración y comunicación de aplicaciones.

Recomendaciones

- No es necesario dominar todos los estándares XML, se recomienda concentrarse en aquellos que sean relevantes para las necesidades específicas. Además muchas herramientas ya cumplen con estándares de modo que lo importante es definir como obtener el máximo beneficio.

- Aprovechar esfuerzos impulsados por diversas empresas e instituciones de tecnología de información que promueven el uso de esta tecnología.

- Para explotar realmente las capacidades y oportunidades que ofrece XML se deben mantener separadas la estructura, los datos y las transformaciones.

- Tomar en cuenta la capacidad de XML para generar con poco esfuerzo múltiples productos distintos a partir de una misma fuente de datos.

- Tomar en cuenta como elementos de riesgo los costos asociados con este tipo de desarrollo, por ejemplo, costos de capacitación, de investigación y de diseño (genérico vrs específico); asimismo la madurez, tanto de las herramientas como de los estándares. Otro elemento de riesgo es la seguridad, la cual debe ser incorporada en el desarrollo desde el primer momento.

- Como toda tecnología, XML no es una llave mágica que resuelva todo, por lo que se debe estar atento a identificar situaciones en las cuales puede no ser la herramienta adecuada.

- Futuras aplicaciones deben ser orientadas a incorporar XML de modo que puedan aprovechar toda la infraestructura de desarrollo y comunicación disponibles. Las aplicaciones o herramientas que no lo incorporen corren el riesgo de quedar rezagadas.

Anexo 1: Estándares relacionados con XML

En este capítulo usted encontrará referencias para:

- los principales estándares relacionados con XML
- los vocabularios más conocidos basados en XML

Como se puede observar en la Figura A.1 la familia de estándares derivados de XML y relacionados con XML es muy numerosa. En la figura se agrupan los estándares por área de trabajo, lo que permite ubicar exactamente los estándares propuestos para áreas como gráficos, procesamiento de voz, transformación de documentos, especificación de modelos y descripción de documentos, comercio electrónico y comunicación de componentes entre otros. También muestra el esfuerzo dedicado por la industria al desarrollo de estos estándares y en cierta forma, la madurez (en cuanto a especificación) alcanzada por los mismos.

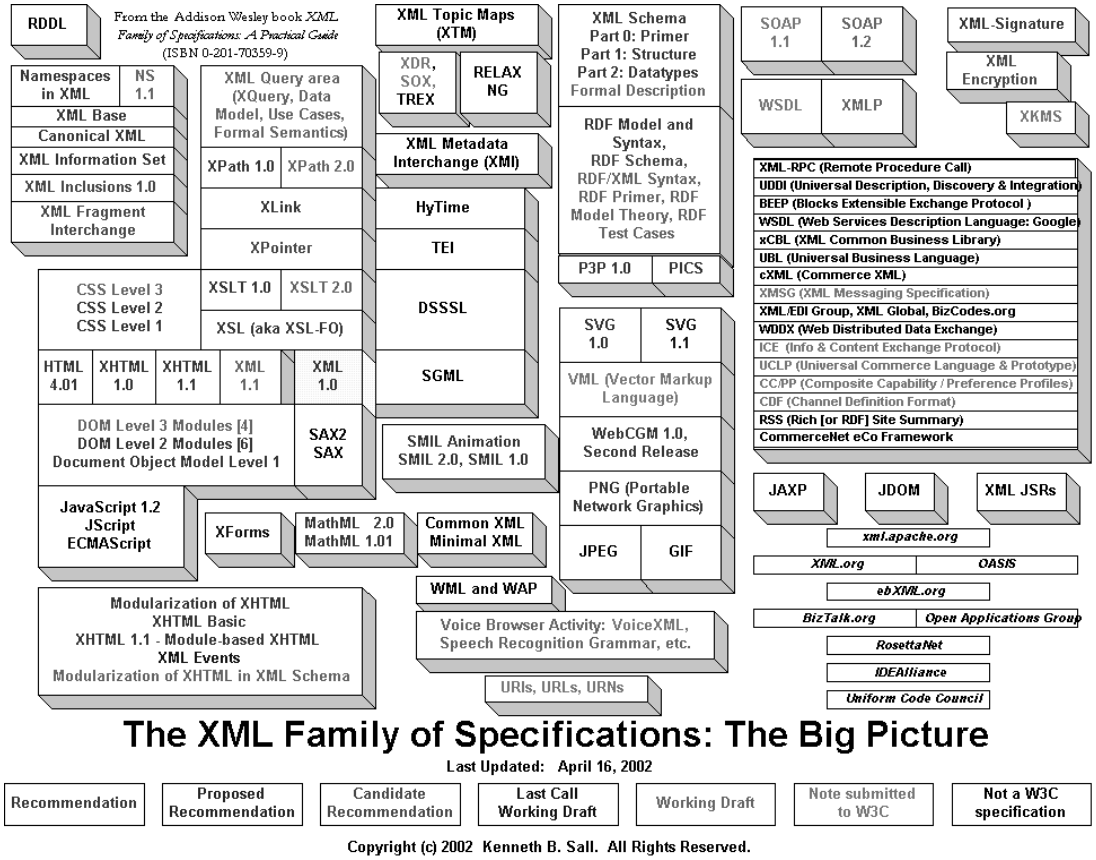
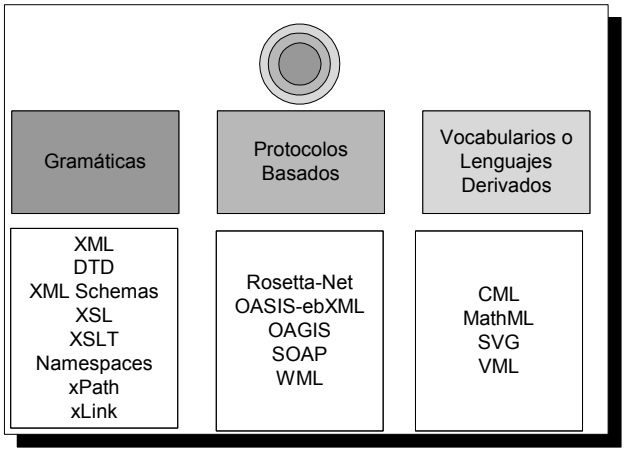


Figura A.1 Familia de especificaciones de XML

Con el objetivo de simplificar la presentación de los estándares, estos pueden separarse en tres categorías o capas, generalmente aceptadas por la industria de Internet, que son **grámatica, protocolo** o **vocabulario** [Dessaux 2001]. La Tabla A.1 muestra los estándares que se detallarán

en este capítulo.

Tabla A.1 Estándares que se detallarán.



Gramática XML

Comprende el núcleo de tecnologías XML definidas por el W3C. Comprende las especificaciones de sintaxis, DTDs, XML Schema, hojas de estilo XSL y transformaciones XSLT, Namespaces y XPath. Estos estándares definen la gramática de XML, sus metadatos y sus lenguajes de transformación.

XML	Especificación	http://www.w3.org/TR/REC-xml
	Descripción	Lenguaje de Etiquetas Extensible. Se considera un formato universal para la definición de estructuras de documentos y datos en la Web
	Tutorial	http://www.w3schools.com/xml/default.asp http://java.sun.com/xml/tutorial_intro.html
	Recursos Importantes	http://www.xml.com/ http://www.oasis-open.org/cover/xml.html http://www.csclub.uwaterloo.ca/u/relander/XML/xml-trt.html http://www.cxml.org

DTD	Especificación	http://www.w3.org/XML/1998/06/xmlspec-v21.dtd
	Descripción	Mecanismo para modelar el contenido de documentos XML y base para validación de documentos
	Tutorial	http://www.zvon.org/xxl/DTDTutorial/General/book.html http://xml101.com/dtd/

	Recursos Importantes	http://www.xml.htmlteacher.com/xmlresources/dtdresources.shtml

Schemas	Especificación	http://www.w3.org/TR/xmlschema-1/ http://www.w3.org/TR/xmlschema-2/
	Descripción	Lenguaje que ofrece facilidades para describir la estructura y restricciones para documentos XML, incluyendo aquellos que usan Espacios de Nombres. Permite además la definición de tipos de datos.
	Tutorial	http://www.w3.org/TR/xmlschema-0/ http://www.xfront.com/xml-schema.html
	Recursos Importantes	http://www.xml.htmlteacher.com/xmlresources/schemaresources.shtml

XSLT	Especificación	http://www.w3.org/TR/xslt http://www.xml.com/pub/a/2001/06/06/xsltspec.html
	Descripción	Lenguaje Extensible de Transformación de Estilos. Lenguaje basado en reglas que transforma un documento XML en un formato diferente.
	Tutorial	http://www.w3schools.com/xsl/ http://vbxml.com/xsl/tutorials/intro/default.asp
	Recursos Importantes	http://www.xsit.com/ http://xml.coverpages.org/xsl.html

XLink	Especificación	http://www.w3.org/TR/xlink/
	Descripción	Lenguaje para describir las relaciones de enlaces que pueden haber entre recursos del Web; en particular, documentos o partes de documentos.
	Tutorial	http://www.xml.com/pub/a/2000/09/xlink http://www-106.ibm.com/deeveloperworks/xml/library/x-xlink/
	Recursos Importantes	http://www.w3.org/XML/Linking http://www.wdvl.com/Authoring/Languages/xLink/

XPointer	Especificación	http://www.w3.org/TR/xptr
	Descripción	Mecanismo para direccionar las estructuras internas de un documento XML.
	Tutorial	http://www.zvon.org/xxl/xpointer/tutorial/OutputExamples/ xpointer_tut.html

	Recursos Importantes	http://webdesign.about.com/od/xpointer/

Namespaces	Especificación	http://www.w3.org/TR/REC-xml-names/
	Descripción	Mecanismo para calificar nombres de elementos y atributos usados en documentos XML asociándolos a espacios de nombres identificados por medio de referencias URI.
	Tutorial	http://www.zvon.org/xxl/NamespaceTutorial/Output/
	Recursos Importantes	

XSL	Especificación	http://www.w3.org/Style/XSL/
	Descripción	Lenguaje de Hojas de Estilo Extensible. Lenguaje para la creación de hojas de estilo. Describe cómo se transforman y formatean datos XML antes de presentarlos al usuario o enviarlos a otro sistema.
	Tutorial	http://www.zvon.org/HTMLonly/XSLTutorial/Books/Book1/bookInOne.html
	Recursos Importantes	http://www.xml.htmlteacher.com/xmlresources/xslresources.shtml

Protocolos Basados en XML

Esta capa comprende los estándares abiertos del W3C. Estos protocolos estpecifican la forma estándar de definir y estructurar el intercambio de mensajes basados en XML entre aplicaciones y sistemas. Se encuentran en esta capa RosettaNet (OASIS-ebXML), Especificación de Integración de Grupos de Aplicaciones Abiertas (OAGIS), Protocolos XML, Protocolo Simple de Acceso a Objetos (SOAP). Algunos de estos protocolos son de dominio específico mientras otros son de propósito general.

Rosetta-Net	Especificación	http://xml.coverpages.org/RNIF-Spec020000.pdf
	Descripción	Consorcio de grandes compañías de Tecnologías de Información, componentes electrónicos, semiconductores, manufactura, telecomunicaciones y logística, que busca crear e implementar estándares abiertos y de amplio uso para el comercio electrónico.
	Tutorial	http://e-docs.bea.com/wli/docs81/tptutorial/intro.html

	Recursos Importantes	http://www.rosettanet.org http://xml.coverpages.org/rosettaNet.html

ebXML	Especificación	http://www.ebxml.org/specs/ebBPSS.pdf
	Descripción	El lenguaje ebXML (Electronic Business using eXtensible Markup) es un protocolo de negocios que permite a compañías realizar negocios en el Internet.
	Tutorial	
	Recursos Importantes	http://e-docs.bea.com/wli/docs81/tptutorial/intro.html

OAGIS	Especificación	OAGIS 8.0 http://www.openapplications.org/downloads/oagis/loadform.htm
	Descripción	Open Applications Group Interface Specifications. Este grupo ha definido un conjunto de especificaciones y estándares, basados en XML, que aseguran la interoperabilidad para aplicaciones empresariales, entre compañías. Esta especificación incluye plantillas para la descripción de documentos, y definiciones para transacciones y procesos de negocio para varios escenarios de e-business
	Recursos Importantes	http://www.openapplications.org/

SOAP	Especificación	http://www.w3.org/TR/SOAP/
	Descripción	Simple Object Access Protocol. Provee un medio para que programas corriendo en un tipo de sistema operativo pueda comunicarse con un programa en el mismo u otro sistema operativo utilizando el protocolo HTTP y XML como mecanismos para el intercambio de información
	Tutorial	http://www.w3schools.com/soap/default.asp http://www.soaprpc.com/tutorials/
	Recursos Importantes	http://www.wdvl.com/Authoring/Languages/XML/Schema.html http://msdn.microsoft.com/webservices/

WAP	Especificación	http://www.openmobilealliance.org/tech/affiliates/wap/wapindex.html
	Descripción	Especificación de un Protocolo de comunicación que estandariza la forma en que los dispositivos inalámbricos pueden utilizarse para acceder Internet
	Tutorial	http://www.w3schools.com/wap/ http://www.palowireless.com/wap/tutorials.asp
	Recursos Importantes	http://webcab.de/wrg/ http://www.palowireless.com/wap/

Vocabularios XML

Los vocabularios XML especifican cómo utilizar XML para una aplicación particular o una industria vertical. Por ejemplo, CML (Chemical Markup Language), derivado de XML, define como se utiliza XML para describir datos de la industria química. Constantemente surgen nuevos vocabularios para nuevos dominios.

BPML	Especificación	http://www.bpmi.org/bpml-spec.esp
	Descripción	El Lenguaje de Modelado de Procesos de Negocio se utiliza para describir procesos de negocios definidos por la organización Business Process Management Initiative. Aplicado al modelado de flujos de trabajo.
	Tutorial	
	Recursos Importantes	http://www.bpmi.org/

CML	Especificación	http://www.xml-cml.org/cml_10.dtd
	Descripción	CML es un lenguaje derivado de XML para la estructuración de información química.
	Tutorial	http://www.ch.ic.ac.uk/omf/cml/doc/tutorial/tutorial.html http://www.zvon.org/xxl/CML1.0/Output/
	Recursos Importantes	http://xml.coverpages.org/cml.html

MathML	Especificación	http://www.w3.org/TR/PR-math
	Descripción	MathML tiene como objetivo facilitar el uso y reutilización de contenido matemático y científico en la Web y en otras aplicaciones.
	Tutorial	http://www.dessci.com/en/support/tutorials/mathml/default.htm
	Recursos Importantes	http://xml.coverpages.org/mathML.html

SVG	Especificación	http://www.w3.org/TR/SVG/
	Descripción	Lenguaje XML para la descripción de gráficos bidimensionales.
	Tutorial	http://www.w3schools.com/svg/default.asp http://www.adobe.com/svg/basics/intro.html
	Recursos Importantes	http://xml.coverpages.org/svg.html http://www.wdvl.com/Authoring/Languages/XML/SVG/DoingIt/ resources.html

WML	Especificación	http://www.wapforum.org/DTD/wml_1.1.xml
	Descripción	Lenguaje que permite que pedazos de texto de páginas Web sean presentados en teléfonos celulares y PDAs vía accesos inalámbricos. WML es parte del protocolo WAP.
	Tutorial	http://www.zvon.org/xxl/WMLTutorial/Output/introduction.html http://www.wirelessdevnet.com/training/WAP/WML.html
	Recursos Importantes	http://www.oasis-open.org/cover/wap-wml.html

WSDL	Especificación	http://www.w3.org/TR/wsdl
	Descripción	Lenguaje de Descripción de Servicios Web.
	Tutorial	http://www.w3schools.com/wsdl/default.asp http://www.xmlspy.com/manual/wsdltutorial.htm
	Recursos Importantes	http://www.freeprogrammingresources.com/wsdl.html http://xml.coverpages.org/wsdl.html

WSFL	Especificación	http://www-3.ibm.com/software/solutions/webservices/pdf/WSFL.pdf
	Descripción	Lenguaje de Flujo de Servicios Web.
	Tutorial	
	Recursos Importantes	http://xml.coverpages.org/wsfl.html

Existen además otros estándares relacionados con tecnologías específicas, los más importantes se describen en el Anexo #2.

En la siguiente sección se mostrarán ejemplos secciones de documentos de vocabularios basados en XML.

Ejemplos de aplicación de XML

Uno de los principales aspectos de XML que la industria ha explotado es la capacidad de definir otros lenguajes de etiquetas para dominios específicos. En realidad son muchos los lenguajes derivados de XML en el mercado, por ello, se han seleccionado solo algunos de ellos, entre los más conocidos, para mostrar las propiedades y beneficios de este tipo particular de lenguajes.

VML (Vector Markup Language)

El Lenguaje de Etiquetas Vectorial soporta las etiquetas de información gráfica de vectores en la misma forma en que HTML soporta las etiquetas de información textual. Dentro de VML el contenido está compuesto por figuras descritas con líneas y curvas conectadas. El Ejemplo A.1 muestra una estrella y a continuación el código VML usado para generarla.

```
<v:shape style='top: 0; left: 0; width: 250; height: 250'
       stroke="true" strokecolor="black" strokeweight="2" fill="true"
       fillcolor="gray" coordorigin="0 0" coordsize="175 175">
<v:path v="m 8,65
  l 72,65,92,11,112,65,174,65,122,100,142,155,92,121,42,155,60,100
  x e"/>
</v:shape>
```

Ejemplo A.1. Uso de VML, basado en [Matthews 1998]

WML (Wireless Markup Language)

El Protocolo de Aplicaciones Inalámbricas (WAP) es el resultado de un trabajo continuo para definir un estándar industrial para desarrollar aplicaciones sobre redes de comunicaciones inalámbricas. WML (Wireless Markup Language) es un lenguaje de etiquetas basado en XML, utilizado en la especificación de contenidos e interfaces de usuarios para dispositivos de poca capacidad como teléfonos celulares y radiolocalizadores. WML está diseñado con las limitaciones de este tipo de dispositivos, que incluyen:

1. Pantallas pequeñas y facilidades de entrada de datos limitadas;
2. Conexiones de redes con ancho de banda reducido;
3. Memoria y recursos computacionales limitados.

En el siguiente ejemplo se muestra una aplicación WML muy sencilla que involucra una interacción con el usuario, quien puede escoger entre un conjunto pequeño de opciones.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC
                "-//WAPFORUM//DTD WML 1.1//EN"
       "http://www.wapforum.org/DTD/wml_1.1.xml">

<wml>

<card id="card1" title="Reservacion Comida">
<do type="accept" label="Respuesta">
  <go href="#card2"/>
</do>
<p>
<select name="tiempo">
  <option value="DESA">Desayuno</option>
  <option value="ALMU">Almuerzo</option>
  <option value="CENA">Cena</option>
</select>
</p>
</card>

<card id="card2" title="Respuesta">
<p>
Su respuesta es: $(tiempo)
</p>
</card>

</wml>
```

```
card1 se vería:

-- Reservacion Comida ---

 Desayuno     •
 Almuerzo     O
 Cena         O

Respuesta


y card2 se vería:

----- Respuesta ---------

 Su respuesta es: DESA
```

Ejemplo A.2 Código WML

SMIL

El Lenguaje de Integración de Multimedios Sincronizados (SMIL) permite la creación de presentaciones audiovisuales interactivas sencillas. Normalmente se utiliza en presentaciones ricas en multimedios que integran videos, sonidos o audio, texto o cualquier otro medio tal como se muestra en el Ejemplo A.3

```
<smil>
<head>
  <meta name="title" content="SMIL Wrapper"/>
  <layout>
    <root-layout background-color="black"
          height=" 315"
           width=" 325"/>
    <region id="videoregion"
          background-color="black"
          top=" 5"
          left=" 5"
          height=" 240"
          width=" 320"/>
    <region id="textregion"
          background-color="black"
          top=" 255"
          left=" 5"
          height=" 60"
          width=" 320"/>
  </layout>
</head>
<body>
  <par>
  <!-- VIDEO -->
    <video src="v-hi.mpg"
           region="videoregion"/>
  <!-- CAPTIONS -->
    <switch>
      <textstream src="v-hi.rt"
          region="textregion"
          system-language="en"
          system-captions="on"
          title="english captions"
          alt="english captions"/>
    </switch>
  </par>
</body>
</smil>
```

Ejemplo A.3 Código SMIL

VoiceXML

VXML, o VoiceXML es una tecnología que permite al usuario interactuar con Internet a través de tecnologías de reconocimiento de voz, utilizando una interfaz de voz o un teléfono.

En el Ejemplo A.4 se muestra una aplicación muy sencilla que solicita al usuario el año de nacimiento, valida si cumple con el requisito de edad de retiro y lo direcciona a la página correspondiente. La etiqueta `prompt` señala el texto que será sintetizado como voz.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
<vxml version="1.0">
  <var name="edadRetiro" expr="65" />
  <form id="checkAge">
     <var name="age" />
     <field name="añoNacimiento" type="date">
       <prompt>Ingrese el año de su nacimiento.</prompt>
     </field>
     <filled>
       <assign name="edad"
               expr="(new Date()).getYear() - añoNacimiento" />
       <if cond="edad >= edadRetiro">
         <goto next="beneficiosRetiro.vxml" />
       <else />
         <goto next="InfopreRetiro.vxml" />
       </if>
     </filled>
  </form>
</vxml>
```

Ejemplo A.4 Código VoiceXML

MathML (Mathematical Markup Language)

MathML es una aplicación XML para la descripción de notación matemática (estructura y contenido). El objetivo de MathML es facilitar que se despliegue información matemática en el navegador justo como se despliega texto [MathML 1998]. El Ejemplo A.5 muestra la fórmula del Teorema de Pitágoras ($x_1^2 + x_2^2 = x_3^2$); el elemento msubsup permite describir un nombre con subíndices y superíndices.

```
<math>
   <mrow>
      <msubsup>
           <mi> x </mi>
           <mrow> <mn> 1 </mn> </mrow>
           <mrow> <mn> 2 </mn> </mrow>
      </msubsup>
      <mo> + </mo>
      <msubsup>
           <mi> x </mi>
           <mrow> <mn> 2 </mn> </mrow>
           <mrow> <mn> 2 </mn> </mrow>
      </msubsup>
      <mo> = </mo>
      <msubsup>
         <mi> x </mi>
         <mrow> <mn> 3 </mn> </mrow>
         <mrow> <mn> 2 </mn> </mrow>
      </msubsup>
  </mrow>
</math>
```

Ejemplo A.5 Código MathML para el teorema de Pitágoras

CML

CML es un enfoque para la administración de información molecular. En palabras simples, es el HTML para las moléculas y un poco más, ya que CML y sus herramientas asociadas permiten la conversión de archivos sin pérdida de semántica en documentos estructurados, incluyendo publicaciones químicas, y provee una ubicación precisa de información dentro de los archivos [CML 2002]. El Ejemplo A.6 muestra cómo describir móleculas a partir de átomos y enlaces.

```
<molecule convention="MDLMol" id="taurine"
          title="TAURINE">
  <date day="23" month="11" year="1995" />
<atomArray>
<atom id="a1">
  <string builtin="elementType">C</string>
  <float builtin="x2">-0.012</float>
  <float builtin="y2">0.0879</float>
  </atom>
<atom id="a2">
  <string builtin="elementType">S</string>
  <float builtin="x2">1.4348</float>
  <float builtin="y2">0.0879</float>
  </atom>
...más átomos
</atomArray>
<bondArray>
   <bond id="b1">
     <string builtin="atomRef">a1</string>
     <string builtin="atomRef">a2</string>
     <string builtin="order">1</string>
     </bond>
    ... más enlaces
</bondArray>
</molecule>
```

Ejemplo A.6 Código CML para la molécula de taurina

Anexo 2: Otras tecnologías relacionadas con XML

COM (Component Object Model)

Modelo para código binario para componentes desarrollado por Microsoft. El modelo COM permite a los programadores desarrollar objetos que pueden ser accesados por cualquier aplicación que cumpla con COM. OLE y ActiveX están basados en COM.

CSS (Cascade Style Sheets)

Desarrollado por la W3C. Es una mecanismo que fue introducido en HTML para darle a los desarrolladores de sitios Web y a los usuarios más control sobre la forma en que se desplegarán las páginas. En las Hojas de Estilo, se define como se verán los diferentes elementos, como encabezados, enlaces, listas, etc. Estos estilos también pueden aplicarse a cualquier página Web.

El término Cascada se deriva de que se pueden aplicar varias hojas de estilo a la misma página Web.

DOM (Document Object Model)

Es la especificación de cómo se representan los objetos de una página Web (textos, imágenes, secciones, encabezados, formularios, etc). También se definen los atributos asociados a estos objetos y cómo se acceden y manipulan. DHTML se basa en el DOM para modificar dinámicamente la apariencia de las páginas Web después de que son descargadas en el navegador del usuario.

RDF (Resource Description Framework)

RDF es una estructura general para la descripción de metadatos de un sitio Web (información sobre la información en el sitio). Provee interoperabilidad entre aplicaciones que intercambian información entendible por la máquina en el Web. Los detalles de la información de RDF incluyen el mapa del sitio, las fechas de las actualizaciones, palabras claves para los motores de búsqueda y los derechos de propiedad intelectual de la página Web.

RDF se basa en XML como sintaxis de intercambio.

DDML Document Definition Markup Language (W3C)

Lenguaje de esquemas para documentos XML. DDML codifica el contenido lógico de la DTD en un documento XML. Esto permite explorar la información de los esquemas y utilizarla con las diferentes herramientas XML disponibles.

DDML es muy simple y, en realidad, provee una base inicial para las implementaciones, está diseñado para tener futuras extensiones como los tipos de datos y la reutilización de esquemas.

DSSSL (Document Style Semantics and Specification Language)
(ISO/IEC 10179:1996)

Es un estándar internacional (ISO/IEC 10179:1996) para especificar la transformación de documentos y el formateo neutral respecto del vendedor y de la plataforma. DSSSL puede utilizarse con cualquier formato de documentos para el cual puede definirse un conjunto de propiedades acorde con los Requerimientos de definición de conjuntos de propiedades (ISO/IEC 10744). En particular, puede utilizarse para especificar la presentación de documentos marcados de acuerdo con el estándar SGML.

DSSSL consta de dos componentes principales: un lenguaje de transformaciones y un lenguaje de estilos.

El lenguaje de transformaciones se utiliza para especificar transformaciones estructurales en archivos fuente SGML. También puede utilizarse para especificar la unión de dos o más documentos, la generación de índices y tablas de contenidos y otras operaciones. Mientras el lenguaje de transformación es una herramienta poderosa para obtener la máxima utilidad de las bases de datos de documentos, las primeras implementaciones de DSSSL se han concentrado en el componente de lenguaje de estilo [Bosak 1999b].

EDI-FACT Electronic Data Interchange For Administration, Commerce and Transport
Comprende los lineamientos de las Naciones Unidas para el Intercambio de Datos Electrónicos para la Administración, Comercio y Transporte. Comprende un conjunto de estándares aceptados internacionalmente, directorios y lineamientos para el intercambio electrónico de datos estructurados y, en particular los relacionados con el mercadeo de bienes y servicios entre sistemas de información computadorizados independientes.

STEP (Standard for the Exchange of Product Model Data)
Paquete de estándares de ISO (TC184/SC4) para el desarrollo de mecanismos para la representación e intercambio de productos digitales de manera neutral. El término de datos de productos denota la totalidad de datos que definen un producto para todas las aplicaciones sobre su ciclo de vida esperado. El objetivo del proyecto es definir una representación de productos que será intercambiada sin pérdida de completitud o integridad.

Este estándar está documentado en volúmenes, llamados Partes, cada uno contiene un elemento de la tecnología STEP.

JAXP (Java API for XML Processing)

API[19] de Java[20] para el procesamiento de documentos XML con DOM, SAX y XSLT. JAXP permite que las aplicaciones analicen y transformen documentos XML independientemente de la implementación de procesamiento de XML particular. Dependiendo de las necesidades de la aplicación, los desarrolladores tienen la flexibilidad de intercambiar entre procesadores XML (parsers de alto rendimiento vrs conservadores de memoria) sin tener que cambiar el código de la aplicación; por esto, los desarrolladores de aplicaciones y herramientas pueden rápida y fácilmente incorporar XML en sus aplicaciones Java para comercio electrónico, integración de aplicaciones, y publicación dinámica en la Web [JAXP].

RDDL (Resource Directory Description Language)

El Lenguaje de Descripción de Directorios de Recursos es una extensión de XHTML básico 1.0 con un elemento adicional que es el recurso. Este elemento sirve como un Xlink al recurso referido, y contiene información legible para humanos de los recursos y de los enlaces (tanto propósito como naturaleza del recurso enlazado). Se utilizan los atributos **xlink:role** y **xlink:arcrole** para indicar la naturaleza y el propósito del recurso respectivamente.

Un Directorio de Recursos es un documento RDDL que provee un paquete de información sobre un Espacio de Nombres de XML, incluyendo:

- Material descriptivo (para humanos) sobre el Espacio de Nombres.
- Directorio de recursos individuales relacionados con el destino, cada entrada de directorio contiene material descriptivo y enlazado con los recursos asociados.
- Ejemplos de recursos individuales relacionados son esquemas, hojas de estilo, código ejecutable diseñado para procesar etiquetas de algún Espacio de Nombres.

El Directorio de Recursos está diseñado para dar servicios como cuerpo de una entidad retornada para desreferenciar a un URI solicitado para identificar un espacio de nombres.

RELAX (Regular Language Description for XML)

RELAX (Descripción para Lenguajes Regulares basados en XML) es una especificación que permite describir lenguajes basados en XML. Por ejemplo el lenguaje XHTML 1.0 puede ser descrito usando RELAX. Este estándar ofrece características adicionales a las DTDs. En particular: las descripciones de RELAX son documentos XML; RELAX puede aprovechar la riqueza de tipos de datos de los Esquemas XML; finalmente, RELAX puede utilizar espacios de nombres. Ofrece las ventajas de aprovechar los tipos de datos de los esquemas y de permitir el desarrollo rápido de herramientas de software debido a que los lenguajes son especificados en XML. [RELAX]

[19] API: Interfaz de programa de aplicación
[20] Java: Lenguaje de programación

SAC (Simple API for Cascading Style Sheets)

La especificación DOM de nivel 2 introduce un Modelo de Objetos para CSS. SAC (API simple para hojas de estilo en cascada) es un estándar que complementa dicho Modelo de Objetos y que incluso puede ser usado para construirlo. SAC provee un API para la sintaxis de CSS pero no accesa la representación interna del motor CSS. Esto permite que nuevos valores de CSS no requieran extensiones de SAC. [SAC]

SAX (Simple Application Programming Interface for XML)

SAX (API simple para XML) es una interfaz que permite escribir aplicaciones que lean datos almacenados en un documento XML. Se trata fundamentalmente de una interfaz Java y es ofrecida por prácticamente todos los "parser" de XML en Java. El nivel de compatibilidad es excelente. Una característica muy peculiar de SAX es que no pertenece a ninguna organización de estándares o consorcios. En particular no tiene relación alguna con la W3C. El procesamiento que establece esta interfaz es del tipo basado por eventos, de modo que los programas están en capacidad de reaccionar cuando el parser encuentra nuevos elementos, o cuando termina un elemento.

Bibliografía

[Barta 2000] Barta, Robert. "*Syndication with JML*" URL: http://www.acm.org/conferences/sac/sac00/Proceed/FinalPapers/WW-12/

[Berghel 1998] Berghel, Hal; Blank, Douglas. "*The WORLD WIDE WEB*" URL: http://www.acm.org/~hlb/publications/web99/web99.html

[Bosak 1997] Bosak, Jon (Sun Microsystems). "*XML, Java, and the future of the Web*" URL: http://www.ibiblio.org/pub/sun-info/standards/xml/why/xmlapps.htm

[Bosak 1998] Bosak, Jon (Sun Microsystems). "*Media-Independent Publishing: Four Myths about XML*" URL: http://www.ibiblio.org/pub/sun-info/standards/xml/why/4myths.htm

[Bosak 1998b] Bosak, Jon (Sun Microsystems). "*XML: The Universal Publishing Format*" SGML/XML Europe'98, URL: http://www.ibiblio.org/bosak/pres/9805eur/sld00000.htm , Paris, mayo 1998.

[Bosak 1999] Bosak, Jon; Bray, Tim. "XML and the Second-Generation Web. The combination of hypertext and a global Internet started a revolution. A new ingredient, XML, is poised to finish the job" Scientific American Vol 280, Num 5 (Mayo 1999) URL: http://www.sciam.com/1999/0599issue/0599bosak.html

[Bosak1999b] Bosak, Jon. "*DSSSL Online Application Profile*" URL: http://www.ibiblio.org/pub/sun-info/standards/dsssl/dssslo/do960816.htm

[Bosworth 1998] Bosworth, Adam. "*Microsoft's Vision for XML*" URL: http://xml.coverpages.org/bosworthXML98.html

[Bray 1998] Bray, Tim. "*Especificación Anotada de XML*" URL: http://www.xml.com/axml/testaxml.htm

[Callahan 2002] Callahan, John R. *Intelligent XML Content Firewalls*, XML Journal, v.3 n.2. Febrero 2002.

[CML 2002] CML.ORG. "*Chemical Markup Language(CMLTM)*" URL: http://www.xml-cml.org/

[Connolly 1997] Connolly, Dan; Khare, Rohit y Rifkin, Adam. "*The Evolution of Web Documents: The Ascent of XML*" URL: http://www.cs.caltech.edu/~adam/papers/xml/ascent-of-xml.html

[Cover 2001] Cover, Robin. "*WAP Wireless Markup Language Specification (WML)*" URL: http://www.oasis-open.org/cover/wap-wml.html

[Cover 2002] Cover, Robin. " *The XML Cover Pages*" URL: http://www.oasis-open.org/cover/

[Dessaux 2001] Dessaux, Claire. "*Understanding XML Standards*" URL: http://www.oracle.com/oramag/oracle/01-mar/o21ind.html?view=print

[Dougherty 1997] Dougherty, Dale. "*Multidimensional Documents: There's a Bright Future Beyond HTML*" URL: http://www.webreview.com/1997/05_16/webauthors/05_16_97_4.shtml

[Freter(a)] Freter, Todd. "*XML: Mastering Information on the Web*" URL: http://www.sun.com/980310/xml/

[Freter(b)] Freter, Todd. "*Beyond Text and Graphics: XML makes Web pages function like applications*" URL: http://www.sun.com/980414/xml/

[Freter(d)] Freter, Todd. "*XML: Document and Information Management*" URL: http://www.sun.com/980908/xml/

[JAXP] Autor desconocido. "*Java API for XML Processing (JAXP)*" URL: http://java.sun.com/xml/jaxp/index.html

[Kamthan 1999] Kamthan, Pankaj. "*XML Namespaces: Universal Identification in XML Markup*" URL: http://tech.irt.org/articles/js193/

[Khare] Khare, Rohit; Rifkin, Adam. "*Capturing the State of Distribuited Systems with XML*" URL: http://www.cs.caltech.edu/~adam/papers/xml/xml-for-archiving.html

[Kuznetsov 2002] Kuznetsov, Eugene. *XML-Aware Networking*, XML Journal, v.3 n.8. Agosto 2002.

[Langham 2003] Langham, Matthew; Ziegeler, Carsten. "*Cocoon:Building XML Applications*", New Riders Publishing, 2003.

[Macherius 1997] Macherius, Ingo. "*XML: a professional alternative to HTML*" URL: http://www.heise.de/ix/artikel/E/1997/06/106/

[Martin 2000] Martin, Didier et all. "*Professional XML*". Wrox Press Ltd, 2000

[Mathews 1998] Mathews; Brian; Lee, Brian, et all. "*Vector Markup Language (VML) Submission to the W3C*" URL: http://www.w3.org/TR/NOTE-VML

[MathML] W3C. "*Mathematical Markup Language (MathML)1.0 Specification"* URL: http://www.w3.org/TR/1998/REC-MathML-19980407/

[Newcomer 2002] Newcomer, Eric. "*Understanding Web Services*: XML, WSDL, SOAP, and UDDI". Addison-Wesley, 2002

[OFX] Organización Open Financial Exchange. "*About OFX*" URL: http://www.ofx.net/ofx/ab_main.asp

[ONUa] ONU. "*UN/EDIFACT Standards*" URL: http://www.c-lab.de/~wolfgang/edi/edifact.html

[ONUb] ONU. "*Standard for the Exchange of Product Model Data*" URL: http://www.c-lab.de/~wolfgang/edi/step.html

[Plotkin] Plotkin, David. "*Building the XML Repository Part 1: The Data Administrator's View of XML*" URL: http://www.intelligenteai.com/XMLRepository/index.htm

[RDDL] Borden, Jonathan; Bray, Tim. "*Resource Directory Description Language (RDDL)*" URL: http://www.openhealth.org/RDDL/

[RELAX] Cover, Robin. "*The XML Cover Pages Regular Language Description for XML (RELAX)*" URL: http://xml.coverpages.org/relax.html

[Rusty 1999] Rusty, Elliote. "*XML Bible*". IDG Books, 1999.

[SAC] W3C. "*Cascading Style Sheets SAC: The Simple API for CSS"* URL: http://www.w3.org/Style/CSS/SAC/

[Salz 2003] Salz, Richard. *XML Within the Enterprise*, XML Journal, v.4 n.1. Enero 2003.

[Standefer 2001] Standefer, Robert. "Enterprise XML Clearly Explained" Morgan Kaufman 2001.

[W3C] W3C. "*Document Definition Markup Language (DDML) Specification, Version 1.0*" URL: http://www.w3.org/TR/NOTE-ddml

[W3C DFD] W3C Document Formats Domain. *Sitio Oficial de SVG de la W3C* URL: http://www.w3.org/Graphics/SVG/Overview.htm8

[W3C 1998] W3C XML Working Group. "*Extensible Markup Language (XML) 1.0 W3C Recomendation 10-Feb-98*" URL: http://www.w3.org/TR/1998/REC-xml-19980210.pdf

[W3C 1998b] Tim Bray (Textuality), Dave Hollander (Hewlett-Packard Company), Andrew Layman (Microsoft). "*Namespaces in XML: World Wide Web Consortium Working Draft 27-March-1998* " URL: http://www.w3.org/TR/1998/WD-xml-names-19980327

[W3C 2001] X*ML Signature specification*. URL:http://www.w3.org/TR/2001/CR-xmldsig-core-20010419.

[W3C 2002] X*ML Encryption specification*:URL: http://www.w3.org/TR/2002/REC-xmlenc-core-20021210.

[W3C 2003] W3C TR Automation, Statistics on recent TR publications. URL: http://www.w3.org/2003/Talks/simo-semwebapp/tr-stats-sample.html, 2003.

[Williamson 2001] Williamson, Heather**. "***XML Manual de Referencia"* McGraw-Hill Osborne Media 2001.