"Modern" XML applications
XML in electronic data interchange, application integration and databases

Patryk Czarnik
Institute of Informatics University of Warsaw

XML and Modern Techniques of Content Management – 2010/11

Electronic data interchange
   Introduction
   Pre-XML solutions
   XML for EDI

Application integration
   Idea
   Web Services

XML in security
   XML Signature
   XML Encryption

XML and databases
   XML support in relational databases
   XML databases
Electronic data interchange (EDI) — motivation

How to interchange data between companies/institutions (B2B)?

▶ paper
▶ electronic data interchange

Standard deployment levels

▶ software developed according to standard from beginning
▶ interface added to legacy system

How to establish EDI protocol?

▶ customer receives (or buys) a tool from provider
▶ smaller partner complies to bigger partner
▶ ad-hoc created conversion tools
▶ standard

EDI standardisation prior to XML introduction

ANSI Accredited Standards Committee X12 sub-group

▶ USA national standard
▶ used mainly in America

EDIFACT

▶ international standard (UN/CEFACT and ISO)
▶ used mainly in Europe and Asia
EDIFACT characteristic

Format

▶ text
▶ hardly readable
▶ tree structure

Predefined dictionaries

▶ 193 message types
▶ 279 segments
▶ 186 elements

(version 08a, 2008)

EDIFACT

EDIFACT message example

UNB+IATB:1+6XPPC+LHPPC+940101:0950+1’
UNH+1+PAORES:93:1:IA’
MSG+1:45’
IFT+3+XYZCOMPANY AVAILABILITY’
ERC+A7V:1:AMD’
IFT+3+NO MORE FLIGHTS’
ODI’
TVL+240493:1000::1220+FRA+JFK+DL+400+C’
PDI++C:3+Y::3+F::1’
APD+74C:0:::6++++++6X’
TVL+240493:1740::2030+JFK+MIA+DL+081+C’
PDI++C:4’
APD+EM2:0:1630::6+++++++DA’
UNT+13+1’
UNZ+1+1’

**EDIFACT structure**

```
Wymiana (interchange)

Wiadomość (message)

Grupa (segment group)

Segment

Złożenie (composite)

Element (data element)
```

```
MEA+WT+AAD+KGM:690+X5'

+KGM:690+

:690
```

**XML EDI**

Idea: use XML as data format for EDI

**Traditional EDI**

- Documents unreadable without specification
- Compact messages
- Centralised standard maintenance
- Changes in format requires software change
- Specialised tools needed

**XML EDI**

- „Self-descriptioning” documents format
- Verbose messages
- “Pluggable”, flexible standards
- Well written software ready to format extensions
- XML-format layer handled by general XML libraries
XML EDI flexibility

Format flexibility

- Structures: choosing, repeating, nesting, optionality
- Format extensions and mixing via namespaces

Applications

- Data interchange between partners’ systems
- Web interface (easy transformation via XSLT)
- Web Services integration

XML EDI standardisation

Framework level

- general rules for all kinds of data
- data of the same kind should be represented in the same way (not to define the same twice)
- example: Electronic Business XML (ebXML).

Industry standards

- SWIFT — banking
- RosettaNet — trade and logistic
- Automotive Industry Action Group — motor industry (mainly American)
- Health Level Seven — health care
- Open Travel Alliance — (people) transport and tourist services
- …
ebXML

- set of specifications defining concepts and methodologies for conducting electronic business via Internet (2001)
- XML used as data format

Electronic Business XML Working Group

- founded in 1999
- more than hundred specialists
- OASIS and UN/CEFACT patronage

ebXML standardisation

- Meta-model:
  - zbiór podstawowych schematów, elementów XML oraz procesów biznesowych,
  - sposób definiowania słowników danych,
  - nie definiuje konkretnych, docelowych komunikatów – mogą one zależeć od konkretnego zastosowania.

- Metainformacje:
  - informacje o wersjach,
  - metadane odpowiadające nagłówkom z istniejących systemów EDI.

- Ramy architektury technicznej:
  - sposoby implementacji repozytoriów, serwisów, itp.,
  - integracja z istniejącymi technologiami EDI.
XML for application integration

- Goal — data interchange between applications
  - applications/modules/components with different internal formats
  - XML as interface
- Usage:
  - client/server communication
  - distributed system nodes
  - components integration
  - configuration of application or components
  - ...

Local and global applications

“Local” integration

- within single project or related projects of single institution
- communication between components
- possibly in distributed architecture
- ad-hoc solutions for given problems
- possibility of using standard

“Global” integration

- services available in Internet for any party
- different parts cooperation
- standardisation required
- most popular standard — Web Services
Web Services

Idea
Web Service — a website for programs (instead of people)

Practice
▶ high-level network protocols (HTTP)
▶ services described (WSDL)
▶ structural messages (XML, SOAP)
▶ possibility of services registration and searching (UDDI)

Web Services — typical applications
▶ Providing data (free or paid):
  ▶ timetables
  ▶ weather
  ▶ stock and currency notes
▶ Services:
  ▶ searching
  ▶ software updates
▶ Business operation between partners
  ▶ booking tickets or hotel rooms
  ▶ ordering (and tracing order status)
  ▶ electronic data interchange
Web Services standardisation

- SOAP (initially “Simple Object Access Protocol”):
  - beginnings: 1998
  - v1.2: W3C Recommendation, June 2003
- Web Services Description Language:
  - v1.1: W3C Note, 2001
  - v2.0: W3C Recommendation, June 2007
- Universal Description Discovery and Integration:
  - OASIS project
  - part of WS-I Basic Profile
- WS-* standards:
  - various standards, usually not W3C:
  - Web Services Interoperability — levels of WS compliance:
    WS-I Basic Profile, Simple Soap Binding Profile, . . . ,
  - WS-Eventing, WS-Addressing, WS-Routing, . . . — IBM documents
- Business Process Execution Language (OASIS) — WS semantics description, programming using WS as building blocks

SOAP — communication protocol

- Underlying transport protocol (HTTP or other)
- Message format (XML)
- Differences to RPC, CORBA, DCOM etc.:
  - data represented in extensible, structural format (XML)
  - data types independent of platform (XML Schema)
  - lower efficiency
SOAP message — general form

▶ XML document for a single message
  ▶ namespace
    http://www.w3.org/2001/12/soap-envelope,
  ▶ main element: Envelope.

▶ Main parts:
  header  optional
  body     required

▶ Restrictions:
  ▶ no DTD (and external entity references)
  ▶ no processing instructions
SOAP header

- **actor** — header receiver identifier (URI), optional
- **mustUnderstand** — must header be understood? (0/1)

W3Schools example

```xml
<?xml version="1.0"?>
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Header>
    <m:Trans
      xmlns:m="http://www.w3schools.com/transaction/"
      soap:actor="http://www.w3schools.com/appml/"
      soap:mustUnderstand="1">234</m:Trans>
  </soap:Header>

  ...

</soap:Envelope>
```

SOAP body

- **remote procedure call**
- **parameters**
- **encodingStyle** — data encoding style (URI)

Request — altered W3Schools example

```xml
<soap:Envelope
 xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
 soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPrice xmlns:m="http://www.w3schools.com/prices">
      <m:Item>Apples</m:Item>
      <m:Currency>PLN</m:Currency>
    </m:GetPrice>
  </soap:Body>

</soap:Envelope>
```
SOAP body

- procedure result
- output parameters

Response — altered W3Schools example

```xml
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <m:GetPriceResponse xmlns:m="http://www.w3schools.com/prices">
      <m:Price>1.90</m:Price>
      <m:Currency>PLN</m:Currency>
    </m:GetPriceResponse>
  </soap:Body>
</soap:Envelope>
```

SOAP — failure message

- standard error code
- short text description
- additional data (XML)

Response with failure message

```xml
<soap:Envelope xmlns:usos="urn:USOS"
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">
  <soap:Body>
    <soap:Fault>
      <soap:faultcode>Receiver</soap:faultcode>
      <soap:faultstring>Data missing</soap:faultstring>
      <soap:faultdetail>Found no student identified with <usos:ind>123</usos:ind></soap:faultdetail>
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```
WSDL — service description

- XML document describing service(s)
  - namespace: http://schemas.xmlsoap.org/wsdl/
  - main element: definitions
- Splitting into parts available

WSDL document components

- **types** type definitions (XML Schema)
- **message** message type definitions
- **portType** set of operations, which have input and output messages
- **serviceType** consists of portType-s
- **binding** service type bound to concrete transport protocol
- **service** concrete service available somewhere

WSDL — messages, operations, port types

W3Schools example

```xml
<message name="getTermRequest">  
  <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">  
  <part name="value" type="xs:string"/>
</message>

<portType name="glossaryTerms">  
  <operation name="getTerm">  
    <input message="getTermRequest"/>  
    <output message="getTermResponse"/>  
  </operation>
</portType>
```
WSDL — SOAP binding

style rpc or document

transport transport protocol (URI)

soapAction SOAP action corresponding to WSDL operation

W3Schools example

```xml
<binding type="glossaryTerms" name="b1">
  <soap:binding style="document"
    transport="http://schemas.xmlsoap.org/soap/http" />
  <operation>
    <soap:operation
      soapAction="http://example.com/getTerm"/>
    <input> <soap:body use="literal"/> </input>
    <output> <soap:body use="literal"/> </output>
  </operation>
</binding>
```

Service registration and discovery

Idea

- service provider registers service
- user searches for service and finds it in registry

Universal Description Discovery and Integration (UDDI)

- available as service (SOAP)
- business category-based directory (“yellow pages”)
- searching basing on service name, description (“white pages”)
- registration and updates for service providers
UDDI — issues

Main issue — who can register?
- anybody — chaos and low reliability
- accepted partners — institution responsible for access policy needed, no such (widely accepted) institution exists

Reality
- UDDI rarely used
- if ever — for “local” SOA-based solutions (intranets)

Service Oriented Architecture

Idea
- services built basing on other services
  - even addition defined as a Web Service :)
- software split into components and layers with WS interfaces between them
- precise specification required (interesting research field. . .)

Critique
- modular, flexible, and scalable solutions
- by the cost of (sometimes) irrational inefficiency and complexity
- use reasonably
XML in security

Main (cryptographic) security aspects

- **confidence** assured by encryption
- **authentication** assured by cryptographic signatures

XML standards related to security

- Signing — XML Signature
- Encrypting — XML Encryption

XML Signature

- **Element Signature** in appropriate namespace
  - cryptographic signature
  - additional information (algorithm etc.)
- Signature added to XML documents
  - externally (*detached signature*)
  - internally (*enveloped signature*)
  - in container enveloping signed data (*enveloping signature*)
- XML Signature capabilities
  - signing fragments of documents
  - signing external resources (identified with URL)
  - multiple signatures in single document
XML Signature — example 1 (detached)

```xml
<Signature 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"/>
    <!- external signed data ->
    <Reference URI="http://example.org/sign-me.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/2000/09/xmldsig#base64"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>60NvZvtdTB+7UnlLp/H24p7h4bs=</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>OsH9A1jTNL...</SignatureValue>
  <KeyInfo>
    <KeyValue><DSAKeyValue>
      <P>imup6lm...</P><Q>xDve3j7...</Q><G>NlugAf...</G>
      <Y>W7dOmH/v...</Y>
    </DSAKeyValue>
  </KeyValue></KeyInfo>
</Signature>
```

Cite: Kazienko, P. Co tam panie w XML-u?, Software 2.0, 6/2003

XML Signature — example 2 (enveloped)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<Document>
  <Content>
    ...
  </Content>
  <ds:Signature>
    <ds:SignedInfo>
      <ds:Reference URI="">
        <ds:Transforms>
          <ds:Transform Algorithm="http://www.w3.org/2000/09/xmldsig
#enveloped-signature"/>
        </ds:Transforms>
      </ds:Reference>
    </ds:SignedInfo>
  </ds:Signature>
</Document>
```
XML Encryption

- Goal: guarantee XML data confidence
- Encryption of whole documents or parts of them

```xml
<purchaseOrder>
  <Order>
    <Item>book</Item>
    <Id>123-958-74598</Id>
    <Quantity>12</Quantity>
  </Order>
  <Payment>
    <CardId>123654-8988889-9996874</CardId>
    <CardName>visa</CardName>
    <ValidDate>12-10-2004</ValidDate>
  </Payment>
</purchaseOrder>
```

XML Encryption — example 1 (fragment encrypted)

```xml
<PurchaseOrder>
  <Order>
    <Item>book</Item>
    <Id>123-958-74598</Id>
    <Quantity>12</Quantity>
  </Order>
  <Payment>
    <CardId>
      <EncryptedData Type='http://www.w3.org/2001/04/xmlenc#Content'
        xmlns='http://www.w3.org/2001/04/xmlenc#'>
        <CipherData>
          <CipherValue>A23B45C564587</CipherValue>
        </CipherData>
      </EncryptedData>
    <CardName>visa</CardName>
    <ValidDate>12-10-2004</ValidDate>
  </Payment>
</PurchaseOrder>
```
XML Support in Databases — Categorisation

Classic (usually relational) database with XML support

- logical structure — relations and references
- additional XML-related features
- used for application integration or storing XML data as part of larger data structures

XML Database

- logical structure — XML document tree
- XQuery (or XPath) as native query language
- natural XML-related features
- used for storing XML data (or structural data easily mapped to XML tree)
XML support in relational databases

Possible functionalities

▶ data import and export in XML format
▶ special treatment of XML data stored in fields
  ▶ XML validation as part of integrity constraints checking
  ▶ XPath or XQuery for querying fields content
▶ XSLT applied to query results

How to store XML data

▶ whole document (fragment) stored in single field
▶ split into prima factors
  ▶ each XML node in separate field
  ▶ tables structure reflects tree structure of XML

Example — XML support in Oracle database

▶ Since Oracle 8i (http://www.oracle.com/xml).
▶ XML parsers
  ▶ for database programming (PL/SQL)
  ▶ or middleware programming (Java, C++)
▶ XML-SQL Utility
  ▶ XML data import and export
▶ XMLType data type
XML-SQL Utility

getXML() function — XML data export

```sql
SELECT xmlgen.getXML('select * from emp') FROM dual;
```

```xml
<rowset>
  <row id="1">
    <empno>10</empno>
    <name>Scott Tiger</name>
    <title>specialist</title>
  </row>
  ...
</rowset>
```

XML support in database engines

Substantial support

- DB2, IBM (since version 9 — pureXML)
- Oracle (since 8i)
- Microsoft SQL Server (od wersji 2000)
- Sybase ASE 12.5

Minimal support

- MySQL (XPath queries over text fields containing XML)
- PostgreSQL???
“XML database”

- **Logical layer**
  - XML document as basic data entity
  - XML schema (or equivalent) as structure definition
  - XQuery (or XPath) as “native” query language
  - document collections

- **Physical layer**
  - not necessarily files with XML documents

- **More than just XML files:**
  - updates interface
  - transactions and concurrent access
  - security

---

**XML:DB**

- Initiative for XML database interfaces specification
- *XML Database API (XAPI)*
  - accessing XML databases from programs
  - resource collections (resource = XML document)
  - reading and writing documents via DOM or SAX
  - pluggable “services”; specified: XPath, transactions, operations on collections
  - last version: 2001

- *XML Update Language (XUpdate)*
  - XML application for updating XML databases
  - inserting, updating and removing nodes
  - XPath used for node addressing
  - last version: 2000
Example (from standard documentation)

```xml
<?xml version="1.0"?>
<xupdate:modifications version="1.0"
  xmlns:xupdate="http://www.xmldb.org/xupdate">
  <xupdate:insert-after select="/addresses/address[1]">
    <xupdate:element name="address">
      <xupdate:attribute name="id">2</xupdate:attribute>
      <fullname>Lars Martin</fullname>
      <born day='2' month='12' year='1974'/>
      <town>Leizig</town>
      <country>Germany</country>
    </xupdate:element>
  </xupdate:insert-after>
</xupdate:modifications>
```

XML databases — overview

<table>
<thead>
<tr>
<th>product</th>
<th>licence</th>
<th>queries</th>
<th>XML:DB API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Xindice</td>
<td>open source</td>
<td>XPath</td>
<td>yes</td>
</tr>
<tr>
<td>BaseX</td>
<td>open source</td>
<td>XPath, XQuery</td>
<td>yes</td>
</tr>
<tr>
<td>eXist</td>
<td>open source</td>
<td>XPath, XQuery</td>
<td>part</td>
</tr>
<tr>
<td>Sedna</td>
<td>open source</td>
<td>XPath, XQuery</td>
<td>yes</td>
</tr>
<tr>
<td>Tamino</td>
<td>paid</td>
<td>XQuery, XPath</td>
<td>part</td>
</tr>
<tr>
<td>Gemfire Enterprise</td>
<td>paid</td>
<td>XQuery, OQL</td>
<td>yes</td>
</tr>
</tbody>
</table>

cite: Wikipedia and providers’ websites