XML Terms and Concepts Primer

Susan Malaika, IBM
malaika@us.ibm.com

NIEM XML & Mobile ID Workshop, Gaithersburg, Sep 2007

Agenda

- What is XML?
- Why XML?
- XML Use Cases
- More on XML
- Well-formed XML documents
- Namespaces
- XPath
- XML Schema
- Various XML Industry Formats
- Summary and Further Reading

Exchanging Data

Let's start with an example...

Here is some data from an ordinary delimited flat file:

47; John Doe; 58; Peter Pan; Database systems; 29; SQL; relational

What does this data mean?

Who can process this data?

An XML Document

XML = eXtensible Markup Language

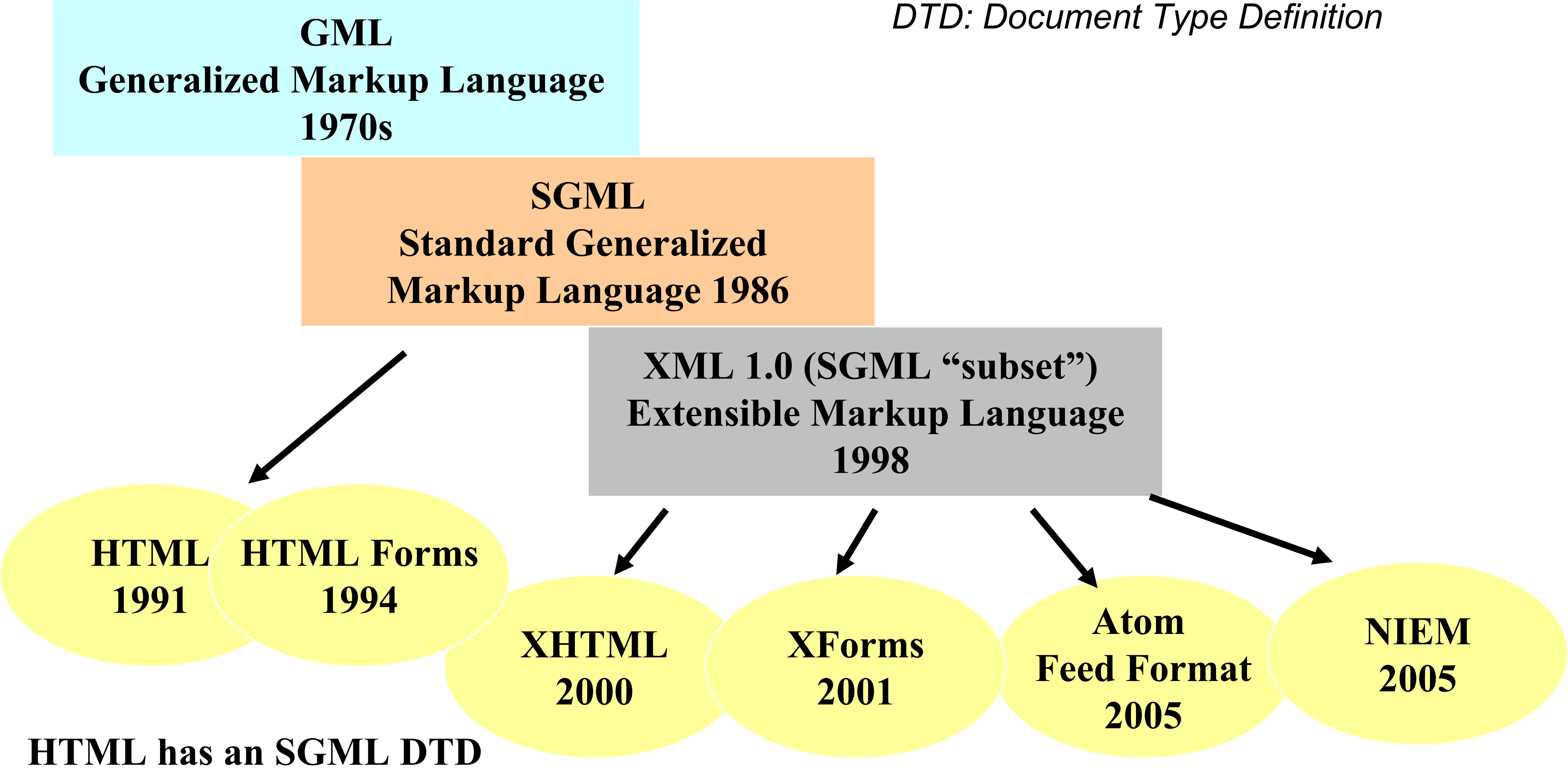
```
<book>
    <authors>
        <author  id="47">John Doe</author>
        <author  id="58">Peter Pan</author>
    </authors>
    <title>Database systems</title>
    <price>29</price>
    <keywords>
        <keyword>SQL</keyword>
        <keyword>relational</keyword>
    </keywords>
</book>
```

XML Document, Message, or Instance

XML: Describes data

XML Time Line

DTD: Document Type Definition



GML
Generalized Markup Language
1970s

SGML
Standard Generalized Markup Language 1986

XML 1.0 (SGML “subset”)
Extensible Markup Language
1998

HTML 1991

HTML Forms 1994

XHTML 2000

XForms 2001

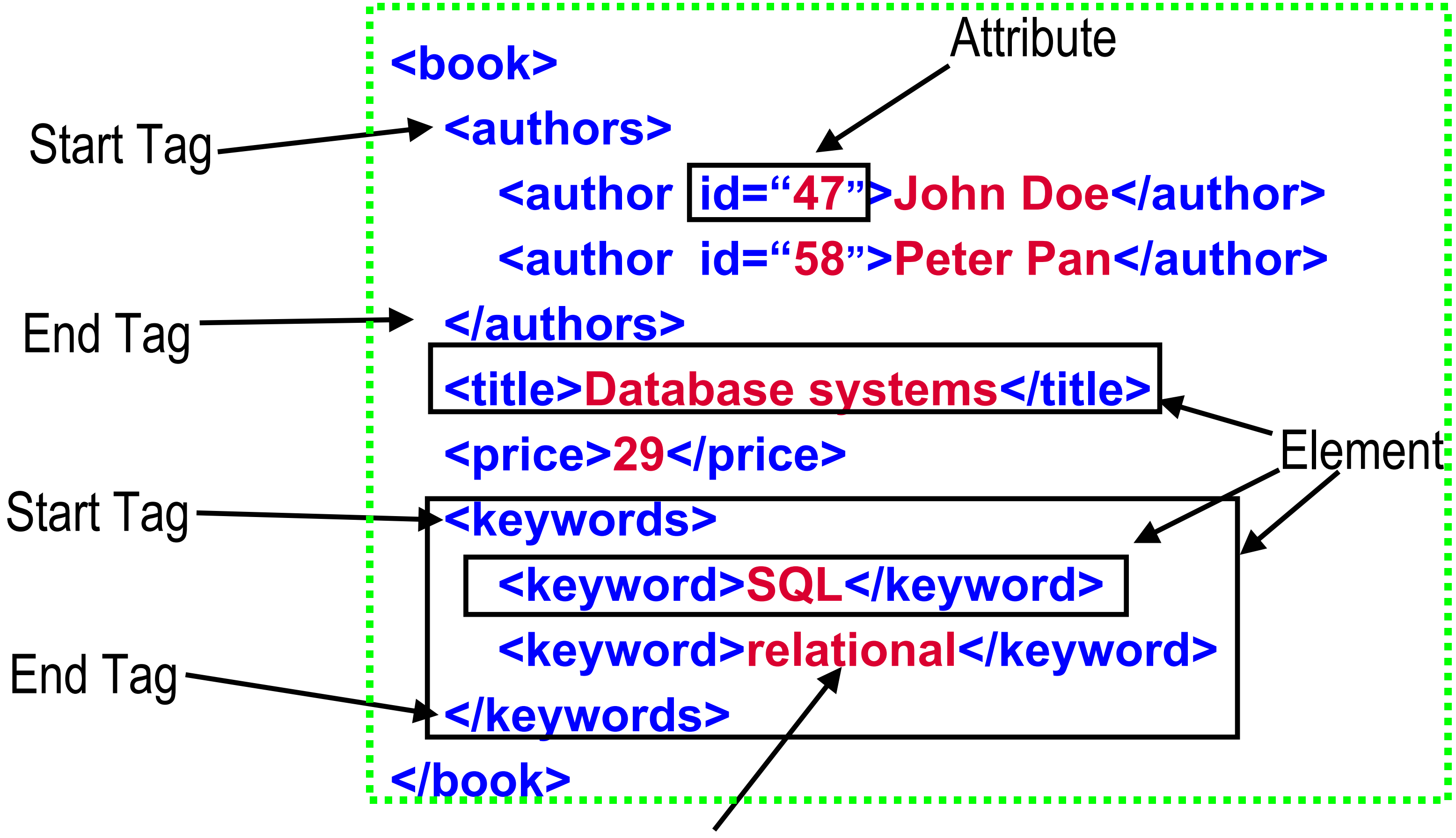
Atom Feed Format 2005

NIEM 2005

HTML has an SGML DTD & is an SGML vocabulary

These formats use XML Schema

Some XML Terminology



```
<book>
  <authors>
    <author id="47">John Doe</author>
    <author  id="58">Peter Pan</author>
  </authors>
  <title>Database systems</title>
  <price>29</price>
  <keywords>
    <keyword>SQL</keyword>
    <keyword>relational</keyword>
  </keywords>
</book>
```

Data or Content

XML: Describes data
HTML: Describes display

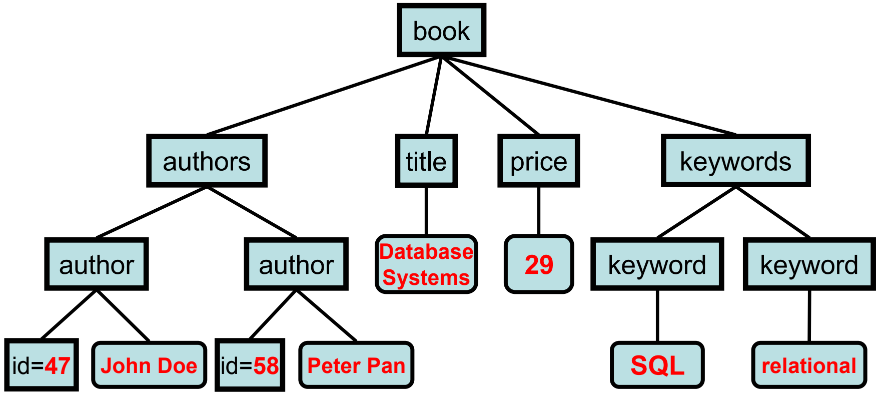
XML Document - 2 Representations

- Two different representations of the **same** hierarchical XML data:

Text

```
<book>
 <authors>
   <author  id="47">John Doe</author>
   <author  id="58">Peter Pan</author>
  </authors>
   <title>Database systems</title>
   <price>29</price>
   <keywords>
     <keyword>SQL</keyword>
     <keyword>relational</keyword>
   </keywords>
</book>
```

Tree



Other representations: Event streams etc.

The XML Document Tree

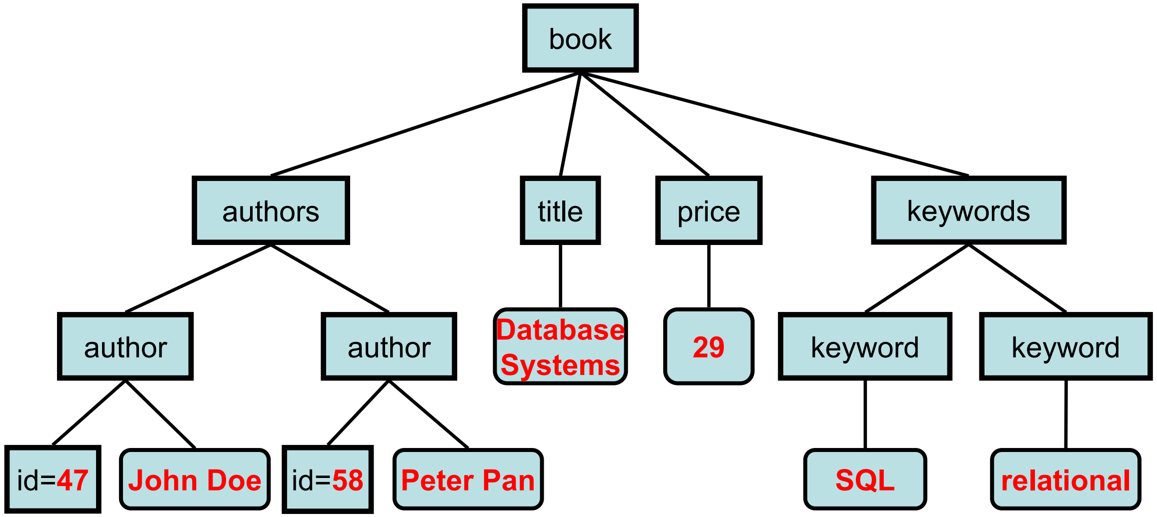
XML Parsers often support:

•DOM: Document Object Model

•SAX: Simple API for XML

```
<book>
     <authors>
          <author  id=“47”>John Doe</author>
          <author  id=“58”>Peter Pan</author>
     </authors>
     <title>Database systems</title>
     <price>29</price>
     <keywords>
          <keyword>SQL</keyword>
          <keyword>relational</keyword>
     </keywords>
</book>
```

XML Parsing



Serialization

Why XML?

- **XML: a notation for data exchange between systems and applications that have not necessarily been formally introduced to each other.**
- **XML: enables the creation of precise descriptions for admissible content -** *you can define the tags*
  - Encoding: Allowable characters and allowable character encodings - to support diverse platforms (operating environments and hardware).
    - Ways to discover the particular encoding
  - Schemas and Constraints: Methods for defining domain specific and general purpose content -
    - Ways to discover, access and process the schemas and constraints
  - Namespaces: Mixing of domain specific content from different sources
    - Ways to use different schemas in a single XML document

More XML Characteristics

- XML has a flexible data model:

  For structured data, semi-structured data, schema-less data,

  For data whose structure evolves

  Easy to extend: define new tags as needed

- XML is self-describing: any XML parser can "understand" it !
- XML is vendor and platform independent
- Easy to "validate" XML, e.g., to check compliance with a schema

  - any XML parser can do it!

- Easy to transform XML documents into other formats

  Plenty of transformation tools

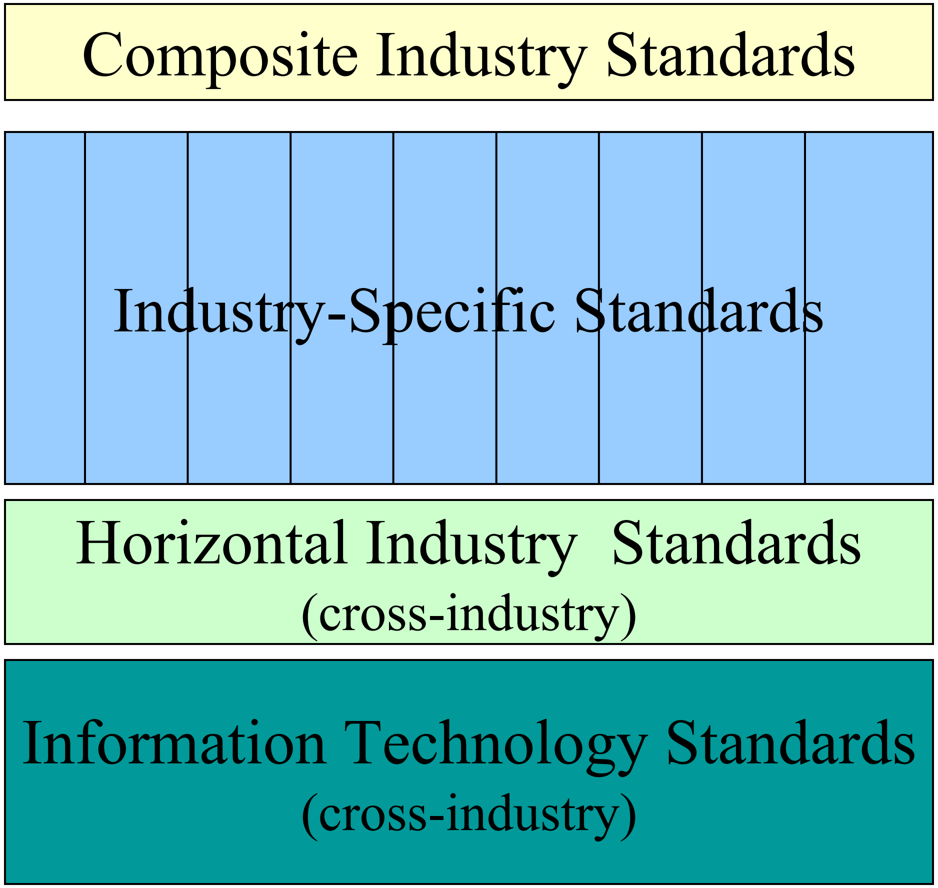
- Easy to share XML between applications, businesses, processes, …

The Success of XML

The success of XML has resulted in the wide availability of :

- General purpose software for processing XML, e.g., XML parsers, XML transformers, XML databases, XML forms processors
- XML content – e.g., XML exchange messages, XML configuration files, XML audit data, XML logs
- XML based IT standards – such as XHTML, and SOAP and WSDL for Web Services
- XML based industry standards – agreed XML schemas for use in specific industries and across industries such as Chemical Markup Language, Healthcare 7, Financila products markup language, Music XML, NIEM

Architecture of Standards



Composite Industry Standards

Industry-Specific Standards

Horizontal Industry Standards
(cross-industry)

Information Technology Standards
(cross-industry)

Examples
IHE, ARTS, RosettaNet
Formats: ACORD, ARTS, FpML, FixML, GJXDM, HL7, HR-XML, MISMO, NewsML, NIEM, MusicXML, STARS, OTA
ebXML, OAGIS, RFID
Formats: XML, Web Services, Atom **Communication:** TCP/IP, HTTP **Addressing:**: URI **Programming**: Java, SQL

XML can be a very good for...

- Data that needs to be exchanged in a variety of ways
  - Example: e-mail, File transfer, Web Services, Fax
- Data that's inherently hierarchical or nested in nature
  - Example: Medical data, Bill-of-materials
- Data sets with sparsely populated attributes
  - Example: FIXML (Financial Information Exchange), FpML (Financial Products Markup Language), Customer profiles, Configuration information
- Data whose structure evolves
  - Example: Frequently changing services/products/processes, product structures, configuration files
- Data that has many variable schemas
  - Example: Data integration, consolidation of diverse data sources
- Data that combines structured & unstructured information
  - Example: Life Sciences, News & Media
- Data that needs to be kept a long time
  - Example; Legal records

Typical XML Use Cases

- Information exchange between applications & organizations
- Message-based transactions, Web Services
- Electronic forms and workflow processing
- Integration of diverse data sources
- Descriptions that change often (configuration files, product structures)
- Content that needs to be human-readable as well as processed by software
- Content that needs to be transmitted in different ways, e.g., file transfer, email, Web Service, feed, fax

- *XML documents as business objects / transaction records (digital signatures, auditing, regulatory compliance)*
- *XML as a flexible data model (for multi-values, hierarchical and complex data)*

Who uses XML? Everybody!

Financial		
ACORD	XML for Insurances	http://www.acord.org/standards/lifexml.aspx
FIXML	Financial Information eXchange protocol	http://www.fixprotocol.org/cgi-bin/Spec.cgi?menu=4
FPML	Financial Product ML	http://www.fpml.org/spec/index.asp
FUNDSML	Funds Markup Language	http://www.funds-xml.org/html/download.htm
XBRL	eXtensible Business Markup Language	http://www.xbrl.org/r
Life Sciences		
AGAVE	Architecture for Genomic Annotation, Visualization and Exchange	http://www.lifecde.com/products/agave/
BSML	Bioinformatic Sequence Markup Language	http://www.bsml.org/resources/default.asp
CML	Chemical Markup Language	http://www.xml-cml.org/
Publication etc.		
SportML	Sport Markup Language	http://www.sportsml.com/specifications.php
NewsML	News Markup Language	http://www.newsml.org/pages/spec_main.php
XBITS	XML Book Industry Transaction Standards	http://www.xmlbits.org/docs.asp
XPRL	eXtensible Public Relations Language	http://www.xprl.org/
Other		
LandML	Land Development Markup Language	http://www.landxml.org/spec.htm
MODA-ML	Middleware tOols and Documents to Enhance the textile/clothing supply chain through xML	http://www.moda-ml.net/moda-ml/repository/schema/V2003-1/default.asp?lingua=en
MatML	Materials Property Data Markup Language	http://www.matml.org/schema.htm
JXDM	Global Justice XML Data Model	http://it.ojp.gov/jxdm/3.0/index.html
ebXML	Electronic Business using eXtensible Markup Language	http://www.ebxml.org/specs/
...	...	...

Use Case 1: Financial Data (FIXML)

- Buying 1000 Shares of IBM Stock..

```
8=FIX.4.2^9=251^35=D^49=AFUNDMGR^56=ABROKER^34=2
^52=20030615-01:14:49^11=12345^1=111111^63=0^64=2003
0621^21=3^110=1000^111=50000^55=IBM^48=459200101^22=
1^54=1^60=2003061501:14:4938=5000^40=1^44=15.75^15=USD
^59=0^10=127
```

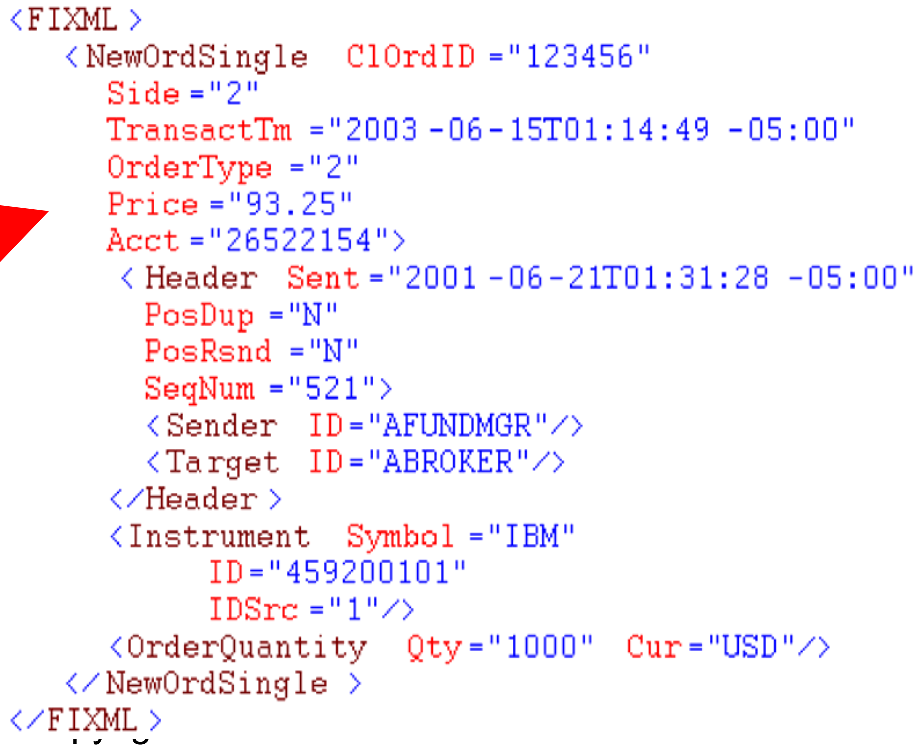
Old FIX Protocol

Financial Information eXchange)

New FIXML

Protocol

- extensible
- lower appl development & maintenance cost



```
<FIXML>
   <NewOrdSingle ClOrdID="123456"
     Side="2"
     TransactTm="2003-06-15T01:14:49 -05:00"
     OrderType="2"
     Price="93.25"
     Acct="26522154">
      <Header Sent="2001-06-21T01:31:28 -05:00"
        PosDup="N"
        PosRsnd="N"
        SeqNum="521">
        <Sender ID="AFUNDMGR"/>
        <Target ID="ABROKER"/>
     </Header>
     <Instrument Symbol="IBM"
           ID="459200101"
           IDSrc="1"/>
     <OrderQuantity Qty="1000" Cur="USD"/>
   </NewOrdSingle >
</FIXML>
```

See http://www.fixprotocol.org/

Use Case 2: newsML – News Data

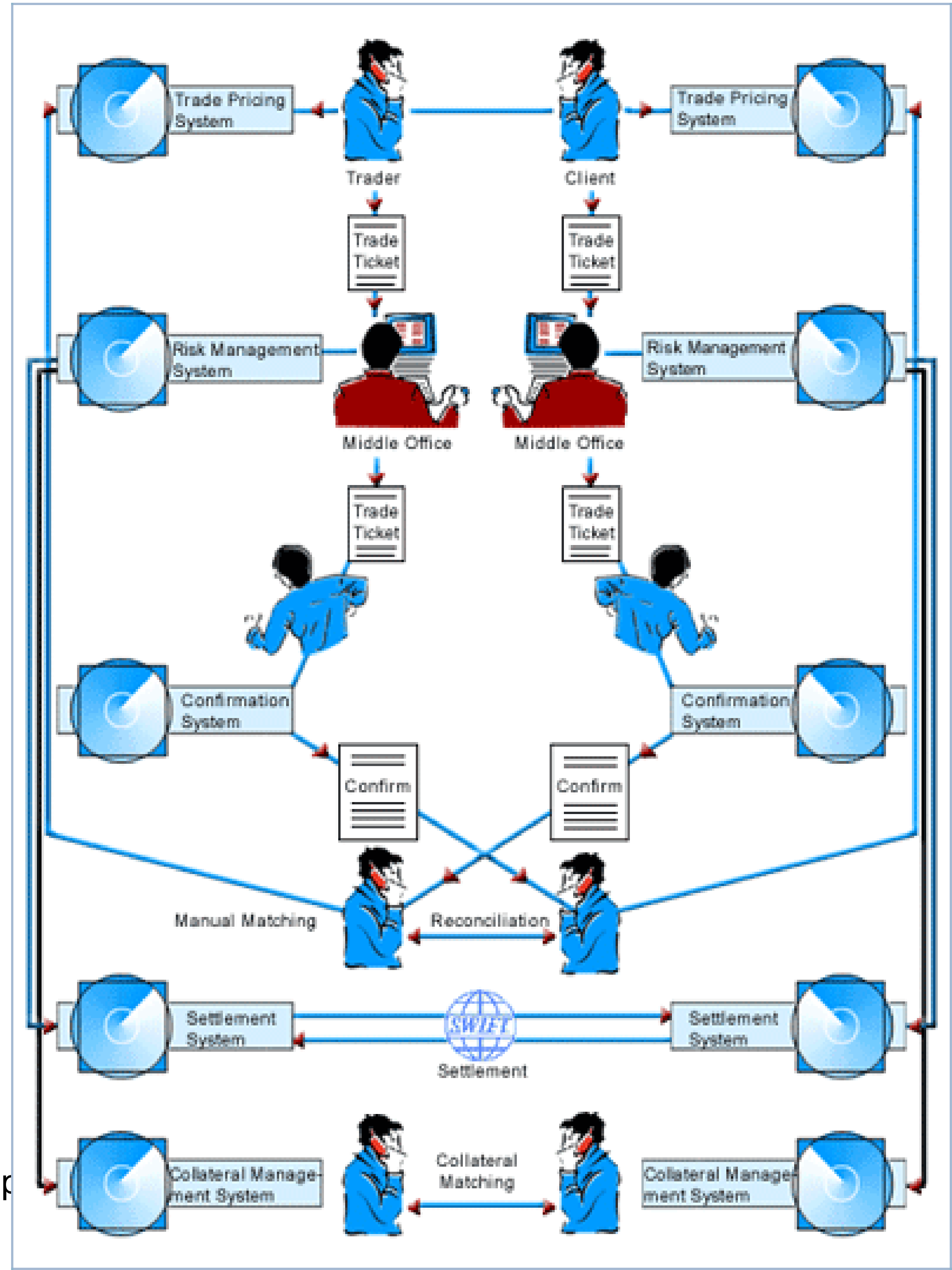
- Worldwide News exchange in XML format
  - between news agencies, publishers, media companies, news editing systems, etc.
- Typically semistructured data
  - some structured data items (date, time, location,...)
  - full text
- Easily searchable
- Easily transformable into any target formatting
- Similar use case: RSS or Atom
  - Online news feeds in XML format
- See http://www.newsml.org/

Use Case 3: FpML - OTC Derivatives Trading

- **F**inancial **p**roducts **M**arkup **L**anguage defined by the ISDA (**I**nternational **S**waps and **D**erivatives **A**ssociation) http://www.fpml.org/
- XML vocabulary for describing derivatives, their trades, and their risks
- Derivatives: risk-shifting agreement, based on any tradable instrument (interest rate, stock, index, currency,…)
- OTC (over-the-counter) derivatives: privately negotiated, no standards, customized contracts
  - Large variety & rapid changes in derivative products & transactions
  - Not manageable in a relational database schema

Derivatives Trading before FpML

- Highly manual, i.e. error-prone and of poor timeliness
- No automated system
- Concern: system not able to handle variety & rapid change
- Concern: too costly to build automated trading system
- Concern: system is obsolete by the time it's implemented
- Solution: XML-based trading system, automated, able to evolve rapidly -> FpML



Benefits of FpML

- Integration of trading services across diverse systems and applications
- Hardware and Software independence
- Lower system implementation & maintenance cost
- Higher trading volumes with higher accuracy
- Increased business opportunities
- Reduced operational risks

Use Case 4: ACORD – Insurance

- ACORD =Association for Cooperative Operations, Research and Development http://www.acord.org/
- Non-profit standards body for insurance data exchange
- 1970: Forms development for property & casualty insurance
- 1980: EDI standards for Property & Casualty industry
- 1996: Standards for Life Insurance
- 2000+:
  - XML-based standards for Property & Casualty, Life, Reinsurance
  - Data and application integration
  - Real-time information exchange for B2B and B2C

Why did ACORD move to XML?

- eBusiness and Internet-based business: connecting back offices, agents, brokers, consumers, etc.
- Diversified & multi-channel distribution
- Streamlined & simplified data transfer
- Straight-through processing of applications & claims
- Cross platform and cross-system data exchange
- Integration of diverse data sources
- Extensible: for hybrid and aggregate insurance products

Well-formed XML Documents

An XML document is well-formed, if:	not well-formed ✗	Well-formed ✓
Has exactly one root element	<b>bla</b> <c>blub</c>	<a> <b>bla</b> <c>blub</c> </a>
Each opening tag is matched by a closing tag	<a><b>bla</a>	<a><b>bla</b></a>
All elements are properly nested	<a><b>bla</a></b>	<a><b>bla</b></a>
Attribute values must be quoted	<a id=15></a>	<a id="15"></a>
Does not use disallowed characters in tags or values	<a> 3<5 </a>	<a> 3&lt;5 </a>
…	…	…

Note: xml header <?xml version="1.0"?> is NOT required for well-formedness.
See http://www.w3.org/TR/REC-xml for full definition.

“Well-formed” or “Valid”?

- An XML document is **well-formed**, if…
  - …it complies with the rules on the previous page
  - i.e. it can be parsed by an XML parser without error
- An XML document is **valid**, if…
  - …it is well-formed AND
  - …it complies with a specific DTD or XML Schema
    - XML Parsers can optionally perform “**validation**”

- DTDs (Document Type Definitions) and XML Schema define a specific XML document structure

Problem: Name Collision

Three different XML elements:

```
<title>Database Administrator</title>

<title>Your majesty</title>

<title>Gone with the wind</title>
```

- Same element name, but different meaning !
- Can result in processing/application errors.
- Need to distinguish between different domains.

Solution: Namespaces

A **prefix** identifies the domain (“namespace”), and distinguishes between duplicate element names

```
<job:title>Database Administrator</job:title>

<person:title>Mr</person:title>

<movies:title>Gone with the wind</movies:title>
```

Namespaces need to be uniquely identified….-> URIs

URI = Universal Resource Identifier

URI Examples:

http://www.ibm.com/db2xml

http://abcdefghijklmn.xyz

- URIs uniquely identify a namespace
- URIs typically *look* like a URL
- URIs are just an identifier, they may to point to a web page, but don't have to !

For more details on URIs see http://www.ietf.org/rfc/rfc2396.txt

Namespace Declaration and URIs

- Example:
  - element name: person
  - namespace URI: http://www.foobar.org
  - namespace prefix: foo

```
<foo:person xmlns:foo="http://www.foobar.org">
  <foo:name>John Doe</foo:name>
</foo:person>
```

The reserved attribute **xmlns** defines namespaces, and (optionally) assigns them to a namespace prefix

The namespace applies to the current element and all sub-elements and attributes that it contains.

Multiple Namespaces

```
<cust:person xmlns:cust=“http://www.foobar.com/customer”>
   <cust:name>John Doe</cust:name>
   <prod:product xmlns:prod=“http://www.foobar.com/product”>
         <prod:name>Thinkpad T40</prod:name>
         <prod:orderdate>2004-11-18</prod:orderdate>
  </prod:product>
</cust:person>
```

Scope of the namespace “prod”

The namespace applies to the current element and all sub-elements and attributes that it contains – unless it’s overridden !

Default Namespaces

A namespace declaration without prefix defines a **default namespace**. The namespace is implicit for all elements/attributes in scope, without using a prefix.

```
<person xmlns=“http://www.foobar.org”>
   <age>45</age>
   <name>
         <first>John</first>
         <last>Doe</last>
  </name>
</person>
```

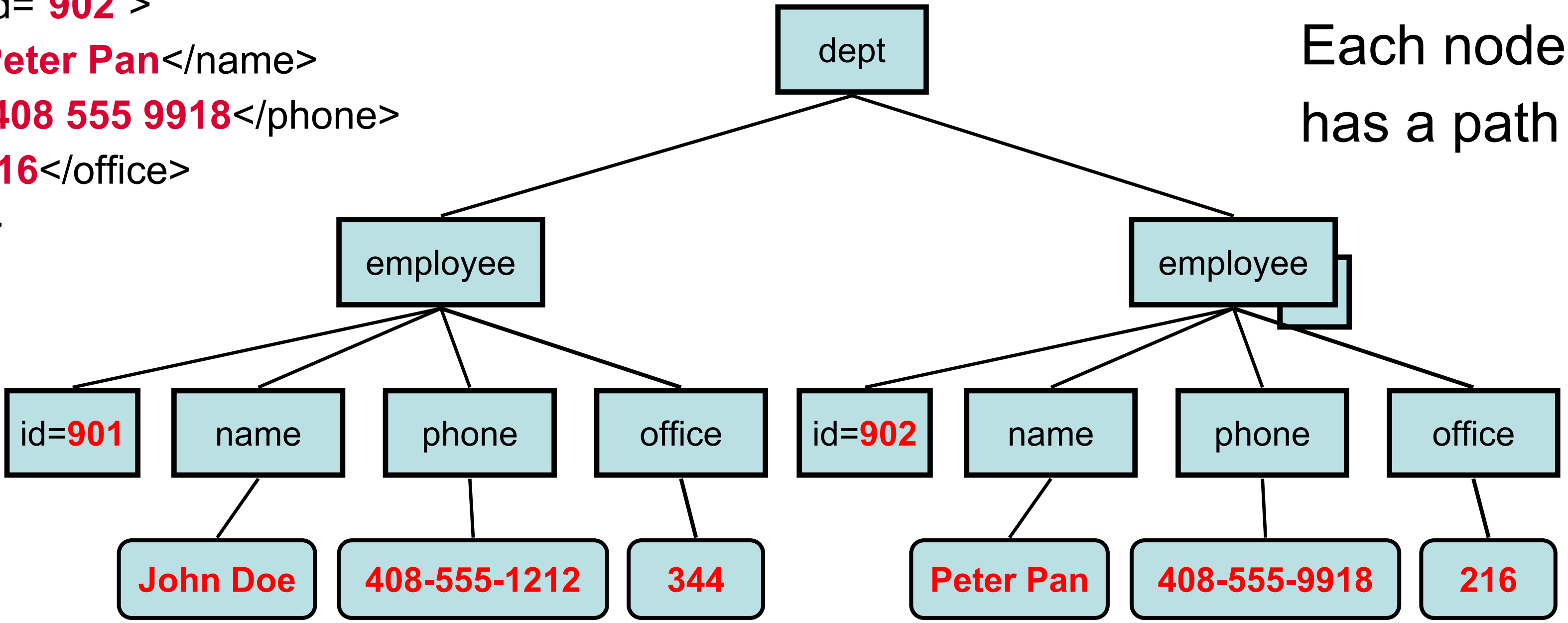
The default namespace applies to the current element and all sub-elements and attributes that it contains.

XPath

```
<dept bldg="101">
   <employee id="901">
        <name>John Doe</name>
        <phone>408 555 1212</phone>
        <office>344</office>
    </employee>
    <employee id="902">
        <name>Peter Pan</name>
        <phone>408 555 9918</phone>
        <office>216</office>
    </employee>
</dept>
```

- *XPath is used in:*
  - *XSLT*
  - *XQuery*
  - *SQL/XML*

/
/dept
/dept/employee
/dept/employee/@id
/dept/employee/name
/dept/employee/phone
/dept/employee/office
(...)



Each node has a path

XPath:
Simple XPath Expressions

```
<dept bldg=“101”>
  <employee id=“901”>
     <name>John Doe</name>
       <phone>408 555 1212</phone>
       <office>344</office>
   </employee>
   <employee id=“902”>
       <name>Peter Pan</name>
       <phone>408 555 9918</phone>
       <office>216</office>
   </employee>
</dept>
```

- Use fully qualified paths to specify elements/attributes
- “@” is used to specify an attribute
- use “text()” to specify the text node under an element

XPath	Result
/dept/@bldg	**101**
/dept/employee/@id	**901** **902**
/dept/employee/name	**<name>Peter Pan</name>** **<name>John Doe</name>**
/dept/employee/name/text()	**Peter Pan** **John Doe**

XPath: Wildcards

- * matches any tag name
- // is the “descendent-or-self” wildcard

```
<dept bldg=“101”>
  <employee id=“901”>
     <name>John Doe</name>
       <phone>408 555 1212</phone>
       <office>344</office>
   </employee>
   <employee id=“902”>
       <name>Peter Pan</name>
       <phone>408 555 9918</phone>
       <office>216</office>
   </employee>
</dept>
```

XPath	Result
/dept/employee/*/text()	**John Doe** **408 555 1212** **344** **Peter Pan** **408 555 9918** **216**
/dept/*/@id	**901** **902**
//name/text()	**Peter Pan** **John Doe**
/dept//phone	**<phone>408 555 1212</phone>** **<phone>408 555 9918</phone>**

XPath: Predicates

- Predicates are enclosed in square brackets […]
- Can have multiple predicates in one Xpath
- Positional predicates: [n] selects the n-th child

```
<dept bldg=“101”>
   <employee id=“901”>
       <name>John Doe</name>
         <phone>408 555 1212</phone>
         <office>344</office>
    </employee>
    <employee id=“902”>
         <name>Peter Pan</name>
         <phone>408 555 9918</phone>
         <office>216</office>
    </employee>
</dept>
```

XPath	Result
/dept/employee[@id=“902”]/name	<name>Peter Pan</name>
/dept[@bldg=“101”]/employee[office >“300”]/name	<name>John Doe</name>
//employee[office=“344” OR office=“216”]/@id	901 902
/dept/employee[2]/@id	902

XPath:
The Parent Axis

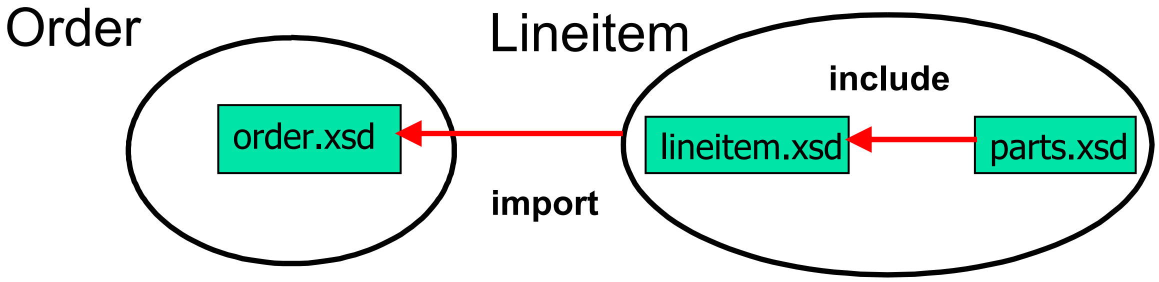
```
<dept bldg="101">
    <employee id="901">
         <name>John Doe</name>
            <phone>408 555 1212</phone>
            <office>344</office>
      </employee>
      <employee id="902">
            <name>Peter Pan</name>
            <phone>408 555 9918</phone>
            <office>216</office>
      </employee>
</dept>
```

- Current context: "."
- Parent context: ".."

XPath	**Result**
/dept/employee/name[../@id="902"]	**<name>Peter Pan</name>**
/dept/employee/office[.>"300"]	**<office>344</office>**
/dept/employee[office > "300"]/office	**<office>344</office>**
/dept/employee[name="John Doe"]/../@bldg	**101**
/dept/employee/name[.="John Doe"]/../../@bldg	**101**

What is an XML Schema?

- Defines structure, content, data types for XML documents
- Consists of 1 or more schema documents, sometimes known as XSDs
- A schema document can define a namespace (optionally)
- Example:
  - 1 XML Schema, 3 Schema Documents, 2 Namespaces



http://www.w3.org/TR/xmlschema-0/

Some XML Schema Characteristics

- Can define data types for elements/attributes
  - Basic types: **integer, date, decimal, string**, etc.
  - User Defined Types, Complex Element Types, etc.
  - Allowed length & patterns for string
  - Supports derived data types
- Detailed occurrence and value range definitions
- XML Schema supports XML **Namespaces**
- A Schema can be composed of multiple schema documents
  - Import/Include of other schemas is supported

Types in XML Schema

Built-In Simple Types:

- string
- boolean
- float
- double
- decimal
- integer
- positiveInteger
- byte
- date
- datetime
- anyType
- ….

Derived Simple Types:

- Restriction of a simple type
  "integer between 5 and 10"
- Union of simple types
  "integer ∪ string"
- Enumerations
- etc.

Complex Types:

- May include elements/attribute definitions
- Can define choice or sequence of elements

XML Schema: Example

```
<xsd:schema    targetNamespace=“http://www.mycompany/products“
                    xmlns:xsd="http://www.w3.org/2001/XMLSchema“>
   <xsd:simpleType name=“PriceType">
      <xsd:restriction base="xsd:decimal">
           <xsd:minInclusive value="0"/>
           <xsd:maxInclusive value=“100000"/>
           <xsd:totalDigits value=“9"/>
           <xsd:fractionDigits value=“3"/>
      </xsd:restriction>
   </xsd:simpleType>
   <xsd:complexType name=“StockPriceType">
       <xsd:sequence>
            <xsd:element name="Ask" type="PriceType"/>
            <xsd:element name="Bid" type="PriceType"/>
            <xsd:element name="P50DayAvg" type="PriceType"/>
      </xsd:sequence>
   </xsd:complexType>
   <xsd:element name=“StockPrice" type=“StockPriceType"/>
</xsd:schema >
```

XML Schema
Namespace

XML Schema: Simple & Complex Types

```
<xsd:simpleType name=“PriceType">
   <xsd:restriction base="xsd:decimal">
     <xsd:minInclusive value="0"/>
     <xsd:maxInclusive value=“100000"/>
     <xsd:totalDigits value=“9"/>
     <xsd:fractionDigits value=“3"/>
   </xsd:restriction>
</xsd:simpleType>
```

PriceType: derived from “Decimal” by defining additional restrictions

```
<xsd:complexType name=“StockPriceType">
    <xsd:sequence>
      <xsd:element name="Ask" type="PriceType"/>
      <xsd:element name="Bid" type="PriceType"/>
      <xsd:element name="P50DayAvg" type="PriceType"/>
   </xsd:sequence>
</xsd:complexType>

<xsd:element name=“StockPrice" type=“StockPriceType"/>
```

Once defined, PriceType can be used multiple times!

A valid instance document

```
<StockPrice>
   <Ask>96.349</Ask>
   <Bid>95.871</Bid>
   <P50DayAvg>89.304</P50DayAvg>
</StockPrice>
```

XML Schema Characteristics for Some Industry Formats [1]

Industry Format	Version	Types	Elements	Attributes	XSD's	Max. XSD size in kB
ACORD (Association for Cooperative Operations Research and Development)	(XMLife) 2.16.01	1369	9378	1275	4	743
ARTS (Association for Retail Technology Standards)	1.0 - 3.0	4825	6305	2011	32	625
CDISC (Clinical Data Interchange Standards Consortium)	00-9-03	98	84	71	1	24
FpML (Financial Products Markup Language)	4.2	686	1867	196	23	130
FIXML (Financial Information Exchange)	4.4	1310	619	2593	41	797
HL7CDA (Clinical Document Architecture)	3	1953	945	477	6	749
IRS1120 (IRS e-File Form 1120)	2006v3.3	3415	11591	2632	600	214
MISMO (Mortgage Industry Standards Maintenance Organization)	2.3 - 2.4	2899	1087	13733	31	2865

XML Schema Characteristics for Some Industry Formats [2]

Industry Format	Version	Types	Elements	Attributes	XSD's	Max. XSD size in kB
MCJE (NIEM) (Minnesota Criminal Justice Event (National Information Exchange Model))	1	415	936	46	7	1661
OTA (OpenTravel Alliance)	2003/5	27293	24893	43141	234	538
STAR (Standards for Technology in Automotive Retail – OAGIS)	5.0.4	5846	77319	625	192	1200
TWIST (Transaction Workflow Innovation Standards Team)	3.1	1016	2314	20	19	154
UBL (Universal Business Language)	2	682	2665	253	43	938
UNIFI (ISO20022 UNIversal Financial Industry message scheme)	1.01 - 2.01	5082	9747	127	71	42
XBRL - GL (Extensible Business Reporting Language - Global Ledger)	2006-10-25	1858	2847	383	45	39

These tables are intended to give an idea of the different design styles for schemas – The schemas may have moved on since these tables was created.

Some XML Design Considerations

- Elements and Attributes
- Document size
- Nesting level
- XML schema features used,
  - e.g., substitution groups, extension elements, complex types
- Schema versioning,
  - e.g., specifying version in schema, in instance document
- Number and structure of XSD files
- Number of message types
- Number and structure of namespaces
- Digital signature provision
- Use of other XML schemas
- Useful complimentary items, such as:
  - forms, e.g., through XForms
  - transformations, e.g., through XSLT
  - queries, e.g., through XQuery
  - constraint check specification e.g., through Schematron

XML Summary [1]

XML is a key technology for…

- …data exchange
- …data integration
- …data evolution and flexibility
- …Web applications & Web services
- …Service Oriented Architectures

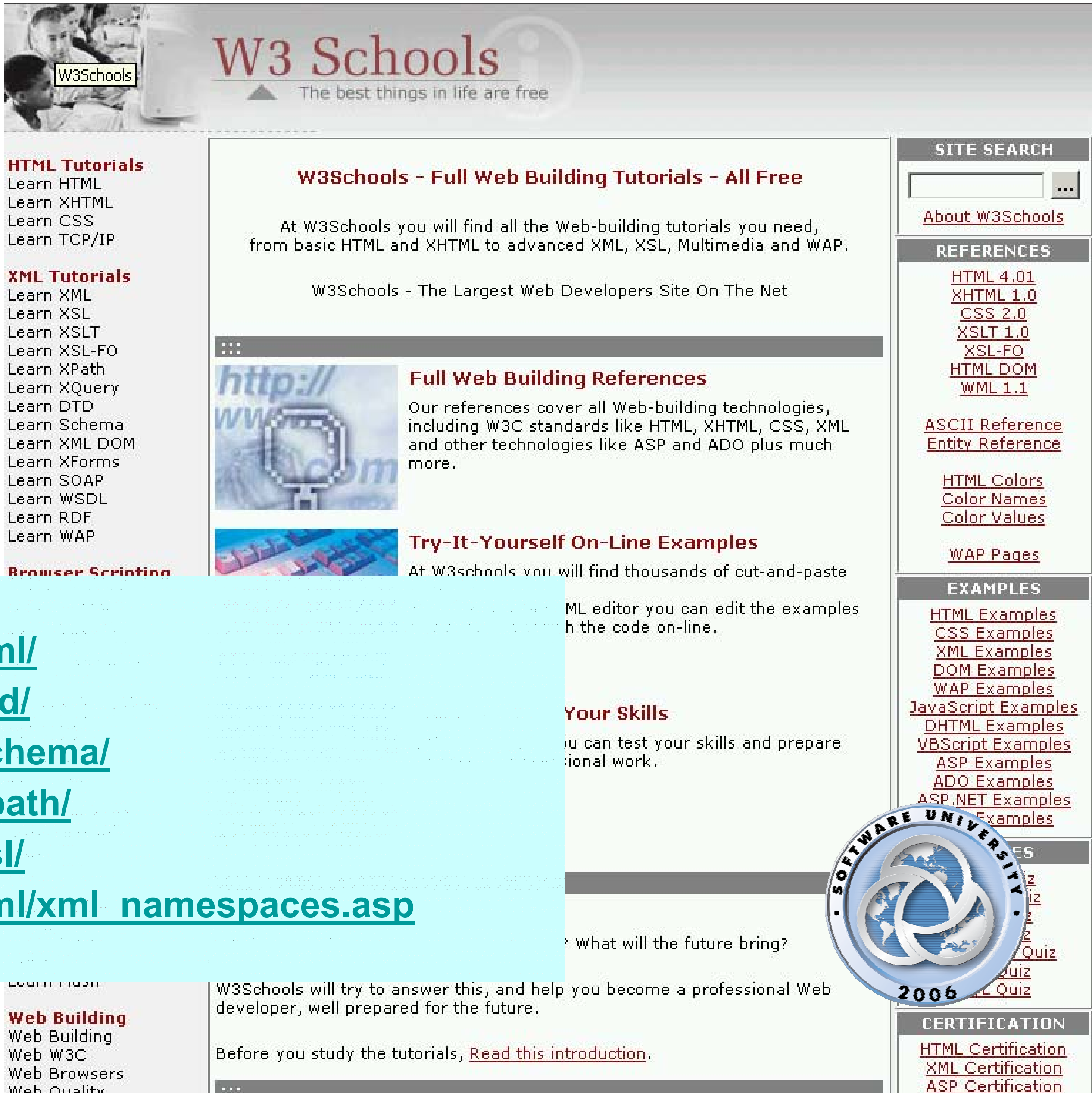
- Software that supports XML is plentiful
  - Hardware that supports XML is also available

XML Summary [2]

- Well-formed XML documents
  - Valid XML documents are well-formed XML that conform to a schema
- Namespaces help:
  - Avoid naming collisions
  - Identify XML vocabularies
- Applying constraints on XML can be achieved through:
  - XML Schemas support data types & namespaces
  - Constraint languages such as Schematron
- XPath, XSLT, XQuery, SQL/XML
  - Search, retrieve, transform modify and shred XML

Further Reading (XML Fundamentals)

Many online tutorials available….



W3 Schools

The best things in life are free

HTML Tutorials
Learn HTML
Learn XHTML
Learn CSS
Learn TCP/IP

XML Tutorials
Learn XML
Learn XSL
Learn XSLT
Learn XSL-FO
Learn XPath
Learn XQuery
Learn DTD
Learn Schema
Learn XML DOM
Learn XForms
Learn SOAP
Learn WSDL
Learn RDF
Learn WAP

Browser Scripting

Learn Flash

Web Building
Web Building
Web W3C
Web Browsers
Web Quality

W3Schools - Full Web Building Tutorials - All Free

At W3Schools you will find all the Web-building tutorials you need, from basic HTML and XHTML to advanced XML, XSL, Multimedia and WAP.

W3Schools - The Largest Web Developers Site On The Net

Full Web Building References

Our references cover all Web-building technologies, including W3C standards like HTML, XHTML, CSS, XML and other technologies like ASP and ADO plus much more.

Try-It-Yourself On-Line Examples

At W3schools you will find thousands of cut-and-paste

ML editor you can edit the examples
h the code on-line.

Your Skills

u can test your skills and prepare
ional work.

What will the future bring?

W3Schools will try to answer this, and help you become a professional Web developer, well prepared for the future.

Before you study the tutorials, Read this introduction.

SITE SEARCH

About W3Schools

REFERENCES

HTML 4.01
XHTML 1.0
CSS 2.0
XSLT 1.0
XSL-FO
HTML DOM
WML 1.1

ASCII Reference
Entity Reference

HTML Colors
Color Names
Color Values

WAP Pages

EXAMPLES

HTML Examples
CSS Examples
XML Examples
DOM Examples
WAP Examples
JavaScript Examples
DHTML Examples
VBScript Examples
ASP Examples
ADO Examples
ASP.NET Examples
Examples

Quiz
Quiz
Quiz

CERTIFICATION

HTML Certification
XML Certification
ASP Certification

http://www.w3schools.com/
http://www.w3schools.com/xml/
http://www.w3schools.com/dtd/
http://www.w3schools.com/schema/
http://www.w3schools.com/xpath/
http://www.w3schools.com/xsl/
http://www.w3schools.com/xml/xml_namespaces.asp

…

September 2007

The Interactive Industry Formats Demo

[1] You can read an article about the demo and bundles:
Get started with Industry Formats and Services with pureXML
http://www.ibm.com/developerworks/db2/library/techarticle/dm-0705malaika/
[2] You can run the demo here:
http://www.alphaworks.ibm.com/tech/purexml
Scroll down on the page to locate the PDF entitled "Getting Started with the Demo" here. Please follow the "Before You Begin" instructions in the PDF. The Interactive Demo (which includes NIEM) can be found by selecting *View Demo*.
[3] You can locate the NIEM Industry bundle by selecting *Download Now*
http://www.alphaworks.ibm.com/tech/purexml
The bundle includes scripts to store, index, validate, and query NIEM in a database

We welcome feedback from the NIEM community on the demo and bundles