XML: related applications and languages

Application areas
Namespaces
XHTML
XML Schema
XLink

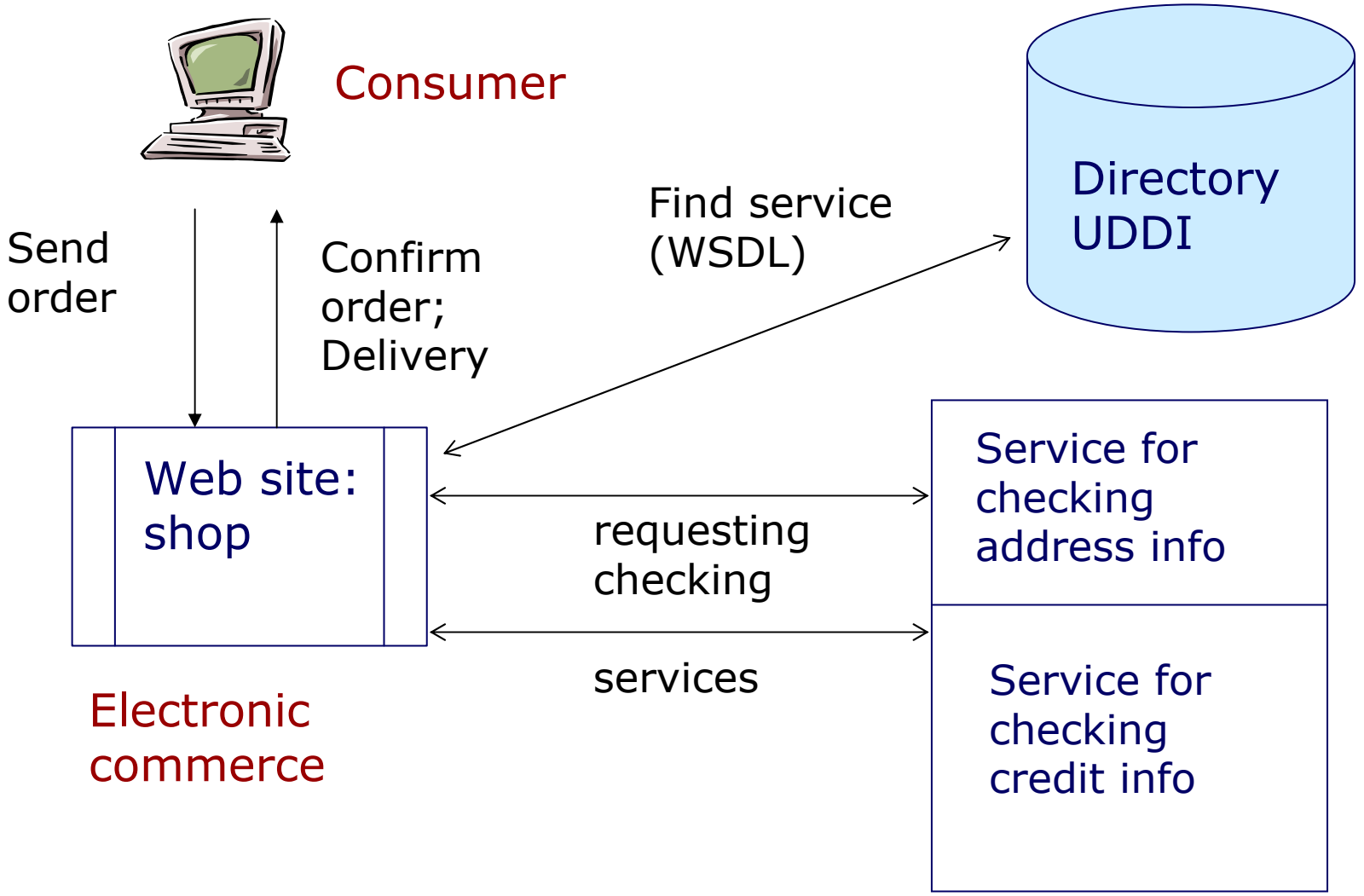
XML application areas

- Documentation:
  - technical documentation: manuals, term banks, spare part catalogs, language versions
  - for example Dublin Core
- Publishing, multichannel publishing
  - documents, DocBook
  - Metadata & semantic web & ontologies
  - same information can be delivered through different media: WWW, PDA, mobile, DVD, print
- User interfaces:
  - Mozilla XUL
  - Microsoft XAML

XML application areas

- Multimedia
  - SVG 1.1; SMIL 2.0
  - Voice XML
  - X3D and x-smiles (Helsinki Univ. of Technology)
- Electronic commerce, EDI (Electronic Data Interchange)
  - ebXML: electronic business XML, UN/CEFACT and OASIS
    - main elements: ebXML storage, CPP – Collaboration Protocol Profile, CPA – Collaboration Partner agreement
  - BizTalk
  - RosettaNet & PapiNet
- Manufacturing: OPC XML-DA schema for plant data exchange based on the SOAP technology and Web Services
- GIS (geographical information systems): GML

Web Services example



XML application areas

- At present ?
- Internal format in browsers
- Microsoft: .NET and internal format for Office
- Ajax and XMLHTTP, Googlemaps

Namespaces - why are they needed?

```
<?xml version="1.0"?>
<person>
<name>
  <title>Mr. President</title>
     <first>Zaphod</first>
<last>Beeblebrox</last>
</name>
<position>President of the Galaxy</position>
  <résumé>
 <html>
   <head><title>Resume of Zaphod Beeblebrox</title>
</head>
      <body>
      <h1> Resume of Zaphod Beeblebrox</h1>
      <p>Zaphod's a great guy, you know?</p>
      </body>
    </html>
  </résumé>
</person>
```

Solution 1:

```
<?xml version="1.0"?>
<pers:person xmlns:pers="http://frogstar.mil/pers"
             xmlns:html="http://www.w3.org/1999/xhtml">
  <pers:name>
    <pers:title>Mr. President</pers:title>
    <pers:first>Zaphod</pers:first>
    <pers:last>Beeblebrox</pers:last>
  </pers:name>
  <pers:position>President of the Galaxy</pers:position>
  <pers:résumé>
    <html:html>
      <html:head><html:title> Resume of Zaphod Beeblebrox
   </html:title></html:head>
      <html:body>
      <html:h1>Zaphod Beeblebrox</html:h1>
      <html:p>Zaphod's a great guy, you know?</html:p>
      </html:body>
    </html:html>
  </pers:résumé>
</pers:person>
```

Namespaces

- A namespace is a purely abstract entity: it’s nothing more than a group of names that belong with each other conceptually
- how to guarantee that the namespace prefix is not used for two different sets of names?
- The internet domain names are unambiguous: URI Uniform Resource Identifier
- problem: the URLs can contain invalid characters
  solution: associate a well-formed prefix with the URI

```
<html:h2 xmlns:html="http://www.w3.org/1999/xhtml">
```

Namespaces, defaults

- Defining a default namespace:

```
<person xmlns="http://frogstar.mil/pers"
        xmlns:xhtml="http://www.w3.org/1999/xhtml">
  <name/>
<xhtml:p>Here something in XHTML</xhtml:p>
</person>
```

- the namespace is defined without a prefix:

  <book xmlns="...">
      <para>A normal paragraph</para>
  </book>

- the default namespace can be changed in any element and any of its descendents

Solution 2:

```
<?xml version="1.0"?>
<person xmlns="http://frogstar.mil/pers"
        xmlns:html="http://www.w3.org/1999/xhtml">
  <name>
    <title>Mr. President</title>
    <first>Zaphod</first>
    <last>Beeblebrox</last>
  </name>
  <position>President of the Galaxy</position>
</position>
  <résumé>
    <html:html>
      <html:head><html:title>Resume of Zaphod Beeblebrox
        </html:title></html:head>
      <html:body>
      <html:h1>Zaphod Beeblebrox</html:h1>
      <html:p>Zaphod's a great guy, you know?</html:p>
      </html:body>
    </html:html>
  </résumé>
</person>
```

Solution 3:

```
<?xml version="1.0"?>
<person xmlns="http://frogstar.mil/pers">
  <name>
    <title>Mr. President</title>
    <first>Zaphod</first>
     <last>Beeblebrox</last>
  </name>
  <position>President of the Galaxy</position>
  <résumé>
    <html xmlns="http://www.w3.org/1999/xhtml">
      <head><title>Resume of Zaphod Beeblebrox
       </title></head>
      <body>
      <h1>Zaphod Beeblebrox</h1>
      <p>Zaphod's a great guy, you know?</p>
      <body>
    </html>
  </résumé>
</person>
```

Namespaces: uses

- Combined documents
  - if you wish to add
    - an HTML table into an XML document
    - how do you avoid mixing up tags and element names?
  - XSL and HTML in the same document
- A DTD owns a namespace where
  - all element names are unique
  - all attribute names for a certain element are unique
  - thus all refrences to the element and its attributes are unique
- document could contain information which is declared in several namespaces

Finding the namespace

- Most standards are on the Web
  - http://www.w3.org/TR/REC-html40
- the namespace recommendation uses URLs as unique identifiers
- the application does not need internet connection, the URL is only a unique character string
- in the document, the prefix is used
  - <X:html xmlns:X="http://www.w3.org/TR/REC-html40">
  - <xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">

XHTML

- XML-DTD for the HTML language
- When moving from HTML to XML you need to double-check that the ”old” HTML file is
  - well-formed
  - valid
  - follows the XHTML specification
- Browser support not consistent in older browsers

XHTML - standards

- XHTML - W3C recommendations 1.0 and 1.1
  based on HTML 4.0
  Working draft 2.0
- XHTML 1.1 is an example of a larger build of the modules, avoiding many of the presentation features.
  - XHTML 1.1 looks very similar to XHTML 1.0 Strict
  - it is designed to serve as the basis for future extended XHTML Family document types
  - modular design makes it easier to add other modules as needed or integrate itself into other markup languages.
  - XHTML 1.1 plus MathML 2.0 document type is an example of such XHTML Family document type.

XHTML - well-formed

- Editing needs for regular HTML
  - add missing end-tags
  - clean out nesting so that the elements are fully within an other
  - spelling of opening and closing tags (<code> and </code>, or <CODE></CODE>)
  - change all characters to lower case
  - add quotes around attribute values
  - change empty tags as empty elements <hr> to <hr/>
  - etc., ...

XHTML - validity

- HTML needs a DOCTYPE declaration
  - if you use only basic elements:
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
  - more permissive:
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
  - frames:
  - <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Frameset//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-frameset.dtd">
- check that it is well-formed

XHTML - other requirements

- HTML file must have <html> as root element
- the namespace has to defined in the root:
  - http://wwww.w3.org/1999/xhtml
- a style sheet processing instruction may be needed

The XHTML example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
     PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">

<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en"
  lang="en">
  <head>
    <title>Virtual Library</title>
  </head>
  <body>
    <p>Moved to
        <a href="http://vlib.org/">vlib.org</a>.</p>
  </body>
</html>
```

XML Schema

- Features in the XML Schema Recommendation include:
  - element types,
  - a simple pattern matching grammar ,
  - defined ordering of sub-elements so that document structure can be tightly controlled,
  - selection between different elements so that documents can share a Schema without having identical structure.

```
<?xml version="1.0"?>
<xsd:schema
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="letter">
</xsd:element>
</xsd:schema>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="header">
    <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="name">
            <xsd:complexType>
            <xsd:sequence>
                    <xsd:element name="title" type="xsd:string"
                    maxOccurs="1"/>
                    <xsd:element name="surname" type="xsd:string" />
                    <xsd:element name="firstname" type="xsd:string" />
            </xsd:sequence>
    </xsd:complexType>
    </xsd:element>
<xsd:element name="address">
<xsd:complexType>
etc..
```

Schema namespaces

```
<?xml version="1.0"?>
<schema xmlns = "http://www.w3.org/2001/XMLSchema"
    xmlns:pers ="http://frogstar.mil/pers"
    targetnamespace = "http://frogstar.mil/pers">
```

In the document:

```
<?xml version="1.0"?>
<person xmlns ="http://frogstar.mil/pers"
    xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation ="http://frogstar.mil/pers people2.xsd"
    version = "1.0">
```

Schema: element types

```
<xsd:simpleType name="userType">
    <xsd:restriction base="xsd:string">
       <xsd:pattern value="U\w{2,6}\d{2}" />
    </xsd:restriction>
</xsd:simpleType>
```

<xsd:complexType>

includes children

<sequence> or <choice> or <all>

Global elements are child elements for the schema element

Local elements are children for another element

Element definitions

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<group name="nimiryhmä">
    <sequence>
        <element name="etunimi" type="string"/ >
        <element name="sukunimi" type="string"/ >
    <sequence>
</group>
<complexType name="nimiryhmä">
    <group ref="target:nimiryhmä"/>
    <attribute name="arvo" type="string"/ >
</complexType>
<element name="Nimet" type="target:nimiryhmä"/>
</schema>
```

Schema: attributes

```
<xsd:complexType name="name">
    <xsd:sequence>
    <xsd:element name="title" type="xsd:string" maxOccurs="1"
    default="Miss"/>
    <xsd:element name="firstname" type="xsd:string" minOccurs
    ="2"/>
    <xsd:element name="surname" type="xsd:string" />
    </xsd:sequence>
    <xsd:attribute name="gender" type="xsd:string"
    default="female"/>
</xsd:complexType>
```

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
    <xsd:element name="header">
    <xsd:complexType>
            <xsd:sequence>
            <xsd:element name="name">
            <xsd:complexType>
              <xsd:sequence>
                        <xsd:element name="title" type="xsd:string"
                        maxOccurs="1"/>
                        <xsd:element name="surname" type="xsd:string" />
                        <xsd:element name="firstname" type="xsd:string" />
              </xsd:sequence>
            </xsd:complexType>
            </xsd:element>
            <xsd:element name="address">
              <xsd:complexType>
                        etc..
```

DTD or Schema?

- XML Schema
  - Follows the XML definition
  - Supports namespaces
  - Allows complex element types
  - Object inheritance

- DTD
  - Can be included in the XML document instance
  - Allows entities
  - A large base of DTDs

GML schema -definition GML.XSD

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
xmlns:sch="http://www.ascc.net/xml/schematron" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="3.0.1">
        <xsd:annotation>
        <xsd:appinfo source="urn:opengis:specification:gml:schema-xsd:gml:v3.0.1">
                gml.xsd
        </xsd:appinfo>
        <xsd:documentation>
        Copyright (c) 2002 OGC, All Rights Reserved. Top level GML schema
        </xsd:documentation>
        </xsd:annotation>
<!-- ========================================================================== -->
<xsd:include schemaLocation="dynamicFeature.xsd"/>
<xsd:include schemaLocation="topology.xsd"/>
<xsd:include schemaLocation="coverage.xsd"/>
<xsd:include schemaLocation="coordinateReferenceSystems.xsd"/>
<xsd:include schemaLocation="observation.xsd"/>
<xsd:include schemaLocation="defaultStyle.xsd"/>
<!-- ========================================================================== -->
</xsd:schema>
```

XML Schema: future

- Large base of existing schemas
- But allows options in definitions:
  - Does not require data type definition – important
  - Version control not defined: paln yourself!
  - Extensible: allows extensions but does not define how
- Supported by many tools like . NET Studio

Creating links in XML

- XML standard way of linking
  - no fixed element for links
  - ID and IDREF attributes identify the element which acts as a link through a namespace
  - only within the same document
- ID values must be unique
- <!ELEMENT chapter (...)>
- <!ATTLIST chapter target ID #REQUIRED>

XLink

- XLink = XML Linking Language
  - to create links within one entity or between two or more targets
- one-direction links (like in HTML)
- extended links
  - bidirectional links
  - links with a type
  - also one-to-many and many-to-many relations
  - links to and from read-only files
  - order of links

XLink, example

Simple link

```
<?xml version="1.0"?>
<ad xmlns:xlink= "http://www.w3.org/1999/xlink">
<p>Never go to this
  <yourlink xlink:type=”simple”
            xlink:href="http://frogstar.mil">
            site!
  </yourlink>
  I told you! </p>
<p> another part of ad  </p>
</ad>
```

Here an arbitrary element is identified using a type attribute in the Xlink namespace

XLink, example 2

Simple link

```
<my:crossReference
  xlink:href="students.xml"
  xlink:role="http://www.ex.com/studentlist"
  xlink:title="Student List"
  xlink:actuate="onLoad"
  xlink:show="embed">
       Current List of Students
</my:crossReference>
```

href = target location
role = location with additional information about target
title = target explanation
actuate - is the link to be followed immediately (onLoad) or later (onRequest)
show - how to show the result (replace, new, embed)

XLink, example 3

Extended link

```
<extendedlink xlink:type="extended">
  <loc      xlink:type="locator" xlink:href="..."
            xlink:label="parent" xlink:title="p1" />
  <loc      xlink:type="locator" xlink:href="..."
            xlink:label="parent" xlink:title="p2" />
  <loc      xlink:type="locator" xlink:href="..."
            xlink:label="child"  xlink:title="c1" />
  <loc      xlink:type="locator" xlink:href="..."
            xlink:label="child"  xlink:title="c2" />
  <loc      xlink:type="locator" xlink:href="..."
            xlink:label="child"  xlink:title="c3" />

  <go       xlink:type="arc" xlink:from="parent"
            xlink:to="child" xlink:actuate="onRequest"
            xlink:show="replace"
            xlink:arcrole=#details.txt"/>
</extendedlink>
```

XLink, example 3 cont.

Extended link, explanation

locator elements (type="locator") - contain one target
  each, indicate remote resources taking part in the link
        href - target location (URI)
        label - link label
        title - link explanation

arc type (type="arc") elements contain the rules for
  traversing from one resource to another
        from
        to
        arcrole   a machine-readable label for a resource