Intelligent Internet Technologies

Lectures 3-4.

XML Markup Technique
(with focus on semantics)

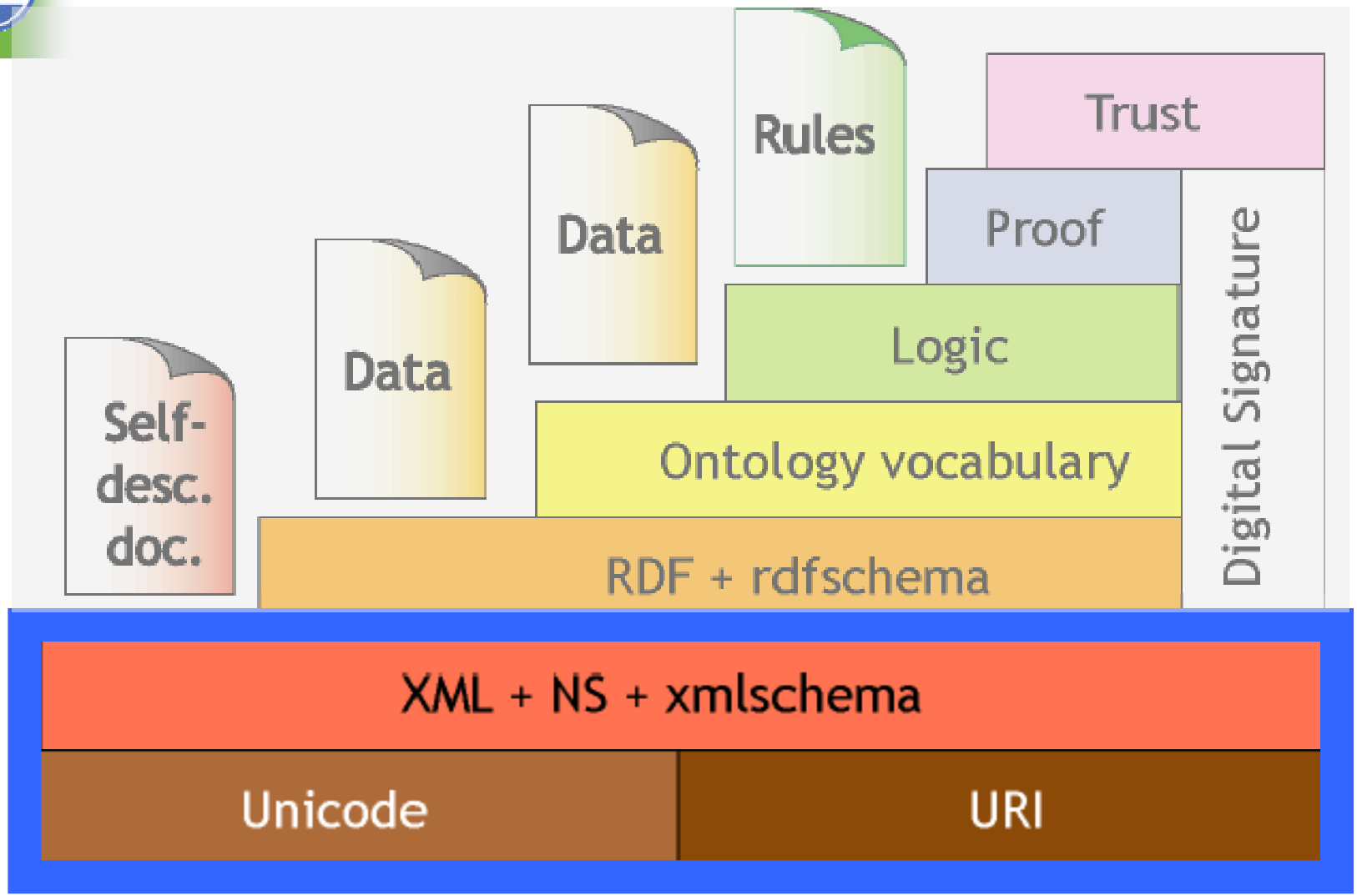
Alexandra V. Vitko

KNURE, AI Department, alexandra_vitko@yahoo.com

Outline

- Unicode
- URI
- XML
- Namespaces

Architecture of Semantic Web



Unicode

What is Unicode?

- The Unicode s tandard defines the universal character set.
- Its primary goal is to provide an unambiguous encoding of the content of plain text, ultimately covering all languages in the world.
- Currently in its fourth major version, Unicode contains a large number of characters covering most of the currently used scripts in the world. It also contains additional characters for interoperability with older character encodings, and characters with control-like functions included primarily for reasons of providing unambiguous interpretation of plain text. Unicode provides specifications for use of all of these characters.

URI

Unified Resource Identifier

What is URI?

- **Uniform Resource Identifiers (URIs)** are short strings that identify resources in the web: documents, images, downloadable files, services, electronic mailboxes, and other resources.
- They make resources available under a variety of naming schemes and access methods such as HTTP, FTP, and Internet mail addressable in the same simple way.

URI versus URL

- URIs are globally scalable unique identifiers.
- A URI is one which does not change. *URIs don't change: people change them.*
- URIs change when there is some information in them which changes.

XML

Extensible Markup Language

XML Technology Includes

- SGML
- XML
- DOM
- XPath
- DTD/XML Schema
- XQL
- CSS
- XSL
- XSLT
- XLink
- XPointer
- etc.

XML Is A W3C Standard

- First official XML specification (1.0) published in February 1998.
- Last version is **1.0 (Third Edition)** W3C Recommendation 04 February 2004.
- Standardization is an important part of what makes XML useful.

XML Is A Markup Language

- A method for putting structured data in a text file.
- A metalanguage – a language used to define your own markup language.
- Uses tags to specify certain rules.
- Used with a processing application that knows how to handle tags.

The “X” in XML

- eXtensible.
- Tags are defined by the person creating the document.
- Tag sets have been developed for specialized topics.
  - Chemistry, math, music, libraries, calendar events, addresses, etc.

What is HTML?

- A simple report-style markup language, supporting:
  - section headings
  - paragraphs
  - tables
  - multimedia
- A mixed collection of tags introduced by Netscape and Microsoft

Why use XML? I know HTML!

- HTML has no means to indicate document structure
- XML separates content and presentation
  - Browser & device specific tags / languages
  - Multi-purpose data
- Difficult to use HTML in applications
- HTML doesn't meet the needs of the modern Internet

XML in 10 Points

-Bert Bos, W3C

1. XML is for structuring data
2. XML looks a bit like HTML
3. XML is text, but isn't meant to be read
4. XML is verbose by design
5. XML is a family of technologies

XML in 10 Points

6. XML is new, but not that new
7. XML leads HTML to XHTML
8. XML is modular
9. XML is the basis for RDF and the Semantic Web
10. XML is license-free, platform-independent and well-supported

Declaration of XML Document

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<mydoc>This is my XML document
</mydoc>
```

- The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document.
- The next line describes the root element of the document

HTML syntax vs. XML syntax

HTML	XML
` `	` `
`<p>text`	`<p>text</p>`
`<p>more text`	`<p>text</p>`
`<b><i>text</b></i>`	`<b><i>text</i></b>`

HTML syntax vs. XML syntax

HTML	XML
`<taBLE>…</TAble>`	`<table>…</table>`
`<body bgcolor=white>`	`<body bgcolor="white">`
`<b>HTML ignores white space</b>`	`<b>XML recognizes white space</b>`
`<i>P&G</i>`	`<i>P&G</i>`

XML Element

<name>Alexandra Vitko</name>

Start tag

Data

End tag

Element

XML Attribute

```
<date format="dd-mm-yy">05-10-99</date>
```

Attribute name Attribute value

Attribute

Things to note:

1. Attributes are included in the start tag.
2. Attributes take this form:

attribute-name="*attribute-value*"

Principles of XML Design: Elements versus Attributes

The never-ending choice

There are some design patterns (???):

Example 1. Principles of

- Document Size
- Ease of Processing
- Flexibility
- Amount of Abstraction

Example 2.

1) Use as many attributes as possible.

2) If it will never ever appear more than once, make it an attribute and the opposite, if something can appear any number of times, it has to become an element.

3) If something represents a seperate entity, make it an element you might want to add attributes later.

4) If something needs to be referenced, make it an element. with an id attribute, couldn't do that with an attribute.

Example 3.

- **Principle of core content:** If you consider the information in question to be part of the essential material that is being expressed or communicated in the XML, put it in an element (data goes in elements, metadata in attributes)
- **Principle of structured information:** If the information is expressed in a structured form, especially if the structure may be extensible, use elements. On the other hand: If the information is expressed as an atomic token, use attributes.
- **Principle of readability:** If the information is intended to be read and understood by a person, use elements. If the information is most readily understood and digested by a machine, use attributes.
- **Principle of element/attribute binding:** Use an element if you need its value to be modified by another attribute.

Common Design Pattern: IDentify classes

Instead of this:

```
<Camera>
    <name>Canon-Sure-Shot-Z155</name>
    <f-stop>4.8-11.7</f-stop>
    <focal-length>37-155mm zoom</focal-length>
    <cost>$318 USD</cost>
</Camera>
```

Design like this:

```
<Camera ID="Canon-Sure-Shot-Z155">
    <f-stop>4.8-11.7</f-stop>
    <focal-length>37-155mm zoom</focal-length>
    <cost>$318 USD</cost>
</Camera>
```

RDF, OWL

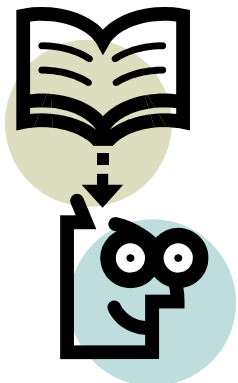
The General Design Pattern

```
<Class ID="instance">
    <property1>data value</property1>
    <property2>data value</property2>
    ...
    <property-n>data value</property-n>
</Class>
```

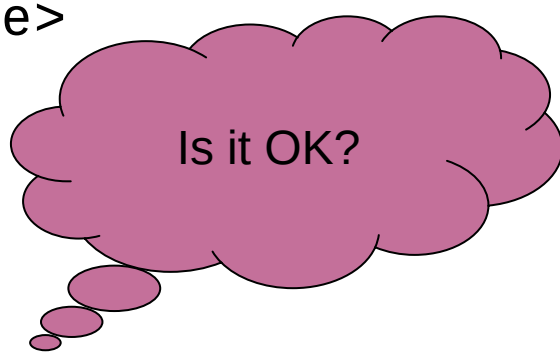
Things to note:

1. Names of Classes by convention begin with uppercase.
2. Names of properties by convention begin with lowercase.

Task for students



- <menu>
  - <menu-item portion="250 mL">
    - <name>Orange Juice</name>
  - </menu-item>
  - <menu-item portion="500 g">
    - <name>Pizza 4 Seasons</name>
  - </menu-item>
- </menu>



Task for students

- <cars>
  - <car1 ID="Audi">
    - <engine>1.8 L</engine>
    - <cost>200 thousands grv.</cost>
    - <ABS>yes</ABS>
  - </car1>
  - <car2 ID="Shevrolet">
    - <engine>1.4 L</engine>
    - <cost>110 000 grv.</cost>
    - <color>blue</color>
  - </car2>
- </cars>

Is it OK?

Task for students

- <people>
  - <manager>
    - <name>Ivan Ivanov</name>
  - </manager>
  - <top>
    - <president>
      - <firstname>Petr</firstname>
      - <last>Petrov</last>
    - </president>
  - </top>
- </people>

Is it OK?

Task for students

- <papka>
  - <risunok>
    - <razmer>120 na 140 pikselej</razmer>
  - </risunok>
- </papka>

Is it OK?

My additional suggestions (for SW)

- Look from the side of machine (search engine, intelligent agent) -> from the side of programmer, who searches information
  - Design in order to easy finding
  - Put homogeneous elements on one level
  - Forget about document size – it should be understandable, readable
  - USE MOST COMMON ENGLISH WORD for tags and attributes!

CData Section

- CDATA section is a character data you want to be left without changes (e.g. program code);

- character data is any string of characters not including the CDATA-section-close delimiter, "]]>".

  <![CDATA[character data **]]>**

Comments in XML

The syntax for writing comments in XML is similar to that of HTML.

<!-- This is a comment **-->**

XML Syntax Summary

- Every tag must be closed, even empty tags
- There is a unique root element
- Proper tag nesting is required
- Tags and content are case sensitive
- Attributes must be quoted
- White space is important
- Elements may not overlap
- Special characters must be escaped

Special Characters

The following characters must be escaped for the XML parser:

<	<
>	>
&	&
‘	'
“	"

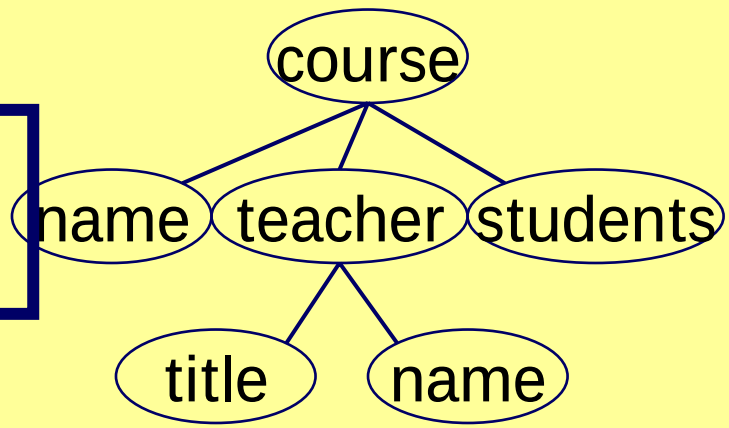
XML Doc as Tree

XML document = labelled tree

node = label + attr/values + contents

```
<course academicyear="...">
      <name>...</name>
      <teacher>
            <title>...</title>
            <name>...</name>
      </teacher>
      <students>...</students>
</course>
```

=



Address Example: external to HTML

External Presentation:

Alexandra Vitko
Lenin av. 14
61166 Kharkov

HTML tags are still presentation-oriented

HTML Markup:

```
<em>Alexandra Vitko</em>
<br>
Lenin av. 14
<br>
<strong>61166 Kharkov</strong>
```

Address Example: HTML to XML

HTML Markup:

```
<em>Alexandra Vitko</em>
<br>
Lenin av. 14
<br>
<strong>61166 Kharkov</strong>
```

XML Markup:

```
<address>
  <name>Alexandra Vitko</name>
  <street>Lenin av. 14</street>
  <town>61166 Kharkov</town>
</address>
```

XML tags are chosen for representation needs

Address Example: XML to External

XML Markup:

```
<address>
  <name>Alexandra Vitko</name>
  <street>Lenin av. 14</street>
  <town>61166 Kharkov</town>
</address>
```

XML stylesheets are, e.g., usable to generate different presentations

External Presentations:

Alexandra Viko
Lenin av. 14
61166 Kharkov

. . .

Alexandra **V**itko
Lenin av. 14
61166 Kharkov

Address Example: XML to XML

XML Markup 1:

```
<address>
  <name>Alexandra Vitko</name>
  <street>Lenin av. 14</street>
  <town>61166 Kharkov</town>
</address>
```

XML Markup 2:

```
<address>
  <name>Alexandra Vitko</name>
  <place>
    <street>Lenin av. 14</street>
    <town>61166 Kharkov</town>
  </place>
</address>
```

XML stylesheets are also usable to transform XML representations

Namespaces

XML Namespaces

XML Namespaces

- Disambiguation of tag and attribute names from different XML applications (“spaces”) through different prefixes

- A *prefix* is separated from the local *name* by a “:”, obtaining *prefix*:*name* tags

Namespace Bindings

- Prefixes are declared with **xmlns:*prefix*** attribute to the prefixed element or one of its ancestors (usually to the root element), and is used by ***prefix*:*name***

- The value of the **xmlns:*prefix*** attribute is a URI, which may or may not point to a description of the namespace's syntax

- An element can use bindings for multiple name-spaces via attributes **xmlns:$prefix_1$,..., xmlns:$prefix_m$**

Namespaces Example: Address Variant

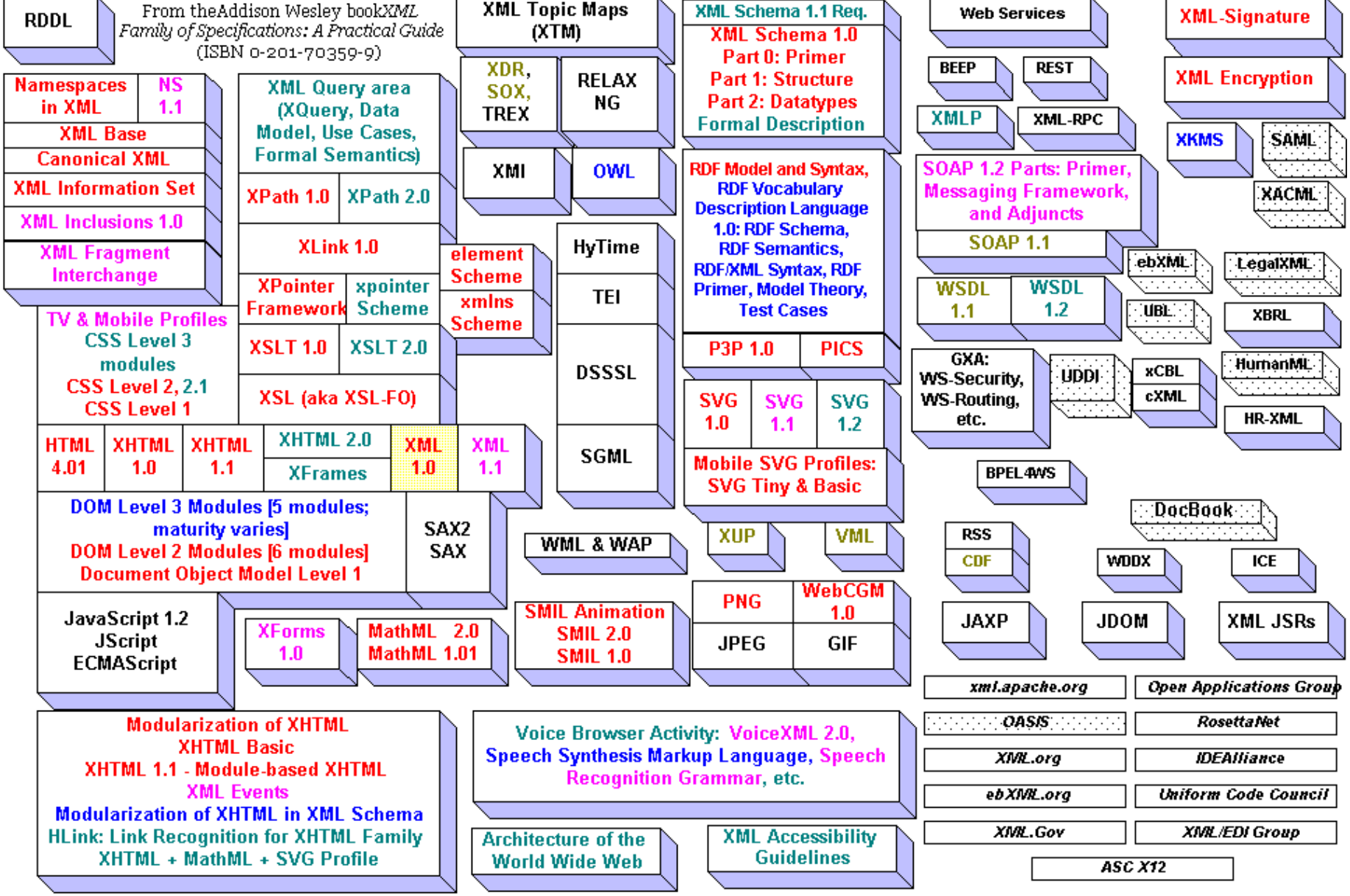
```
<address>
   <name>Alexandra Vitko</name>
   <street>Lenin av.</street>
   <town>61166 Kharkov</town>
   <bill>12.50</bill>
   <phone>0577020214</phone>
   <fax>0577020214</fax>
   <bill>76.20</bill>
</ address>
```

bill is ambiguous tag – bill for post and bill for phone

Two-Namespaces Example

```
<mail:address xmlns:mail="http://www.tnt.com/"
              xmlns:tele="http://www.ukrtelecom.ua/">
  <mail:name>Alexandra Vitko</mail:name>
  <mail:street>Lenin av. 14</mail:street>
  <mail:town>61166 Kharkov</mail:town>
  <mail:bill>12.50</mail:bill>
  <tele:phone>057/7020214</tele:phone>
  <tele:fax>057/7020214 </tele:fax>
  <tele:bill>76.20</tele:bill>
</ mail:address>
```

bill disambiguation through mail and tele prefixes



The XML Family of Specifications: The Big Picture

Last Updated: April 19, 2003

Recommendation	Proposed Recommendation	Candidate Recommendation	Last Call Working Draft	Working Draft	Note submitted to W3C	Not a W3C specification

Advantages of XML for Knowledge Representation

XML offers new general possibilities, from which AI knowledge representation can profit:

(1) Definition of self-describing data in worldwide standardized format

(2) Structured data and knowledge exchange for enterprises in various industries

(3) Integration of information from different sources (into uniform documents)

Key Uses of XML

- Data storage
- Data exchange
- Document publishing

Traditional Data Storage

- Databases
  - Time and cost to create and maintain.
- Flat Files
  - Format is not standardized.
  - Must write your own input/output and validation programs.

XML: Data Storage

- Searching the data is relatively easy.
- Format is standard.
- Standard tools for input/output and validation exist.
- Easy to read files makes debugging easier.

Traditional Data Exchange

- Nonstructural values
  - Data is hard to read/identify.
  - Relationships between data are hard to document.
- Fixed fields
  - Limited to certain field widths.

XML: Data Exchange

- Data is relatively easy to read and edit with a simple text editor.
- Complex relationships like trees and inheritance can be communicated.
- Tags are self-describing, human readable.
- Automatic data validation.

Traditional Document Display

- HTML
  - Content and GUI are mixed.
  - Searching for information in the data is tough.
  - Content is tied to the logic and language of HTML.

XML: Document Display

- Tags are handled by XSL.
  - Instructions for transforming one kind of document to another.
  - Common transformation is XML to HTML
- One XML may be linked to multiple XSL files.
- Separation of content from presentation.

Read More in

- W3C
  - http://www.w3.org/XML/
- XML.org
  - http://xml.org
- O'Reilly.com
  - http://www.xml.com/