W3C Technologies: a Key for Interoperability

Oreste Signore

W3C Office in Italy and ISTI-CNR
Area della Ricerca CNR di Pisa – via G. Moruzzi, 1 – 56124 Pisa (Italy)
e.mail: oreste@w3.org
personal home page: http://www.weblab.isti.cnr.it/people/oreste/

Abstract. In the present application environment, facing the fully decentralized nature of the web, interoperability is both a key success factor and quality issue. .The family of XML technologies is playing a significant role in the present Web. A coherent usage of XML technologies leads to implementation of applications showing significant accessibility, portability and flexibility levels, with reduced costs. In addition, coherence with basic Web principles and World Wide Web Consortium (W3C) long term goals guarantees evolution of applications towards the ambitious Semantic Web objective. In this paper we will discuss the technological and semantic interoperability, showing how W3C technologies can help to implement effective, cost saving applications, and will support the evolution from cross applications interoperability towards a web of meaning.

Introduction

W3C leads the evolution of the Web, pursuing its long term goals. Among the others, Universal Access and Semantic Web show a significant impact upon interoperability. However, we want to stress that we can have two different kinds of interoperability: technological and semantic. In this paper we will discuss the key role played by XML to support *interoperability within applications*, while RDF helps in *cross applications interoperability*.
The paper will firstly give a brief overview of W3C Consortium and of the evolution of the web.
After discussing the issue of interoperability, the paper will show some W3C technologies related to technological interoperability, putting some emphasis on structuring and presenting content rich information. Subsequently, the paper considers the metadata issue, with a brief description of RDF and the Semantic Web stack. Finally, Web Services are presented as a case for interoperability.

The World Wide Web Consortium (W3C)

The *Wide Web Consortium* (W3C) was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has more than 400 *Member organizations* from all over the world and has earned international recognition for its contributions to the growth of the Web.
By promoting *interoperability* and encouraging an open forum for discussion, W3C commits to leading the technical evolution of the Web. In just over seven years, W3C has developed more than fifty *technical specifications* for the Web's infrastructure. However, the Web is still young and there is still a lot of work to do, especially as computers, telecommunications, and multimedia technologies converge. To meet the growing expectations of users and the increasing power of machines, W3C is already laying the foundations for the next generation of the Web. W3C's technologies will help make the Web a robust, scalable, and adaptive infrastructure for a world of information. To understand how W3C pursues this mission, it is useful to understand the Consortium's goals and driving principles.
The W3C's *long term goals* are coherent with the initial motivations that lead to the birth of the web. In summary, they can be synthetically listed as:

- *Universal Access*: To make the Web accessible to all by promoting technologies that take into account the vast differences in culture, languages, education, ability, material resources, and physical limitations of users on all continents
- *Semantic Web*: To develop a software environment that permits each user to make the best

use of the resources available on the Web

- *Web of Trust*: To guide the Web's development with careful consideration for the novel legal, commercial, and social issues raised by this technology

The Web must evolve at a pace unrivalled in other industries: almost no time is required to turn a bright idea into a new product or service and make it available on the Web to the entire world. With an audience of millions applying W3C specifications and providing feedback, W3C concentrates its efforts on three principle tasks:

1. *Vision*: W3C promotes and develops its vision of the future of the World Wide Web. Contributions from several hundred dedicated researchers and engineers and from the entire Web community enable W3C to identify the technical requirements that must be satisfied if the Web is to be a truly universal information space.
2. *Design*: W3C designs Web technologies to realize this vision, taking into account existing technologies as well as those of the future.
3. *Standardization*: W3C contributes to efforts to standardize Web technologies by producing specifications (called "Recommendations") that describe the building blocks of the Web. W3C makes these Recommendations freely available to all.

W3C Recommendations

W3C Recommendations result after a cooperative process, as described in the *Process Document*, where members contribute substantially, and results are public. Figure 1 shows the process, making evident as there are some possible cases of returning to a previous stage. It must be recalled that moving from “Last Call Working Draft” to “Candidate Recommendation” stage implies a “Call for implementation”, and the “Proposed Recommendation” stage is reached after a satisfactory implementation experience. Therefore, we can say that we have both a “proof of the concept” as well as a “proof of implementation”. Hence, W3C Recommendations do not have the risk of being “paper standards”, but are proved to be effective and can be implemented with a reasonable effort.

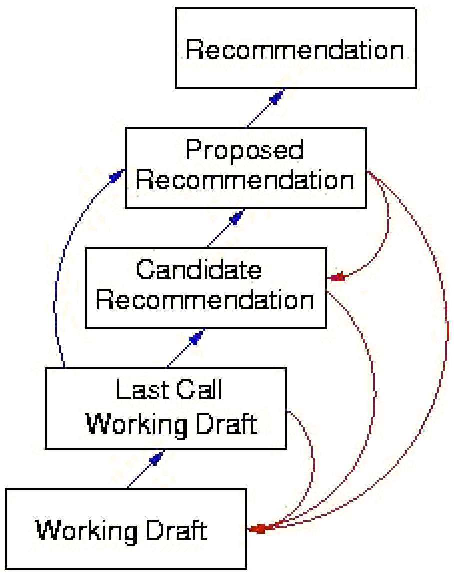


Figure 1 - The W3C Recommendation Track

Even if W3C Recommendations are usually quoted as “web standards” it is evident that W3C is not, officially, a standardization body. However, W3C is a consortium whose members cooperate spontaneously (and often enthusiastically) to define really guidelines that can be implemented and used by the large user community. In addition, W3C maintains very close relationships wit standardization bodies and User Forums.

The Web

Design Principles

The Web is an application built on top of the Internet and, as such, has inherited its fundamental design principles.

- *Interoperability*: Specifications for the Web's languages and protocols must be *compatible* with one another and allow *(any) hardware and software* used to access the Web to work together.
- *Evolution*: The Web must be able to accommodate *future technologies*. Design principles such as *simplicity*, *modularity*, and *extensibility* will increase the chances that the Web will work with emerging technologies such as mobile Web devices and digital television, as well as others to come.
- *Decentralization*: this is without a doubt the *newest* principle and most *difficult* to apply. To allow the Web to "scale" to worldwide proportions while resisting errors and breakdowns, the architecture (like the Internet) must *limit or eliminate dependencies on central registries*.

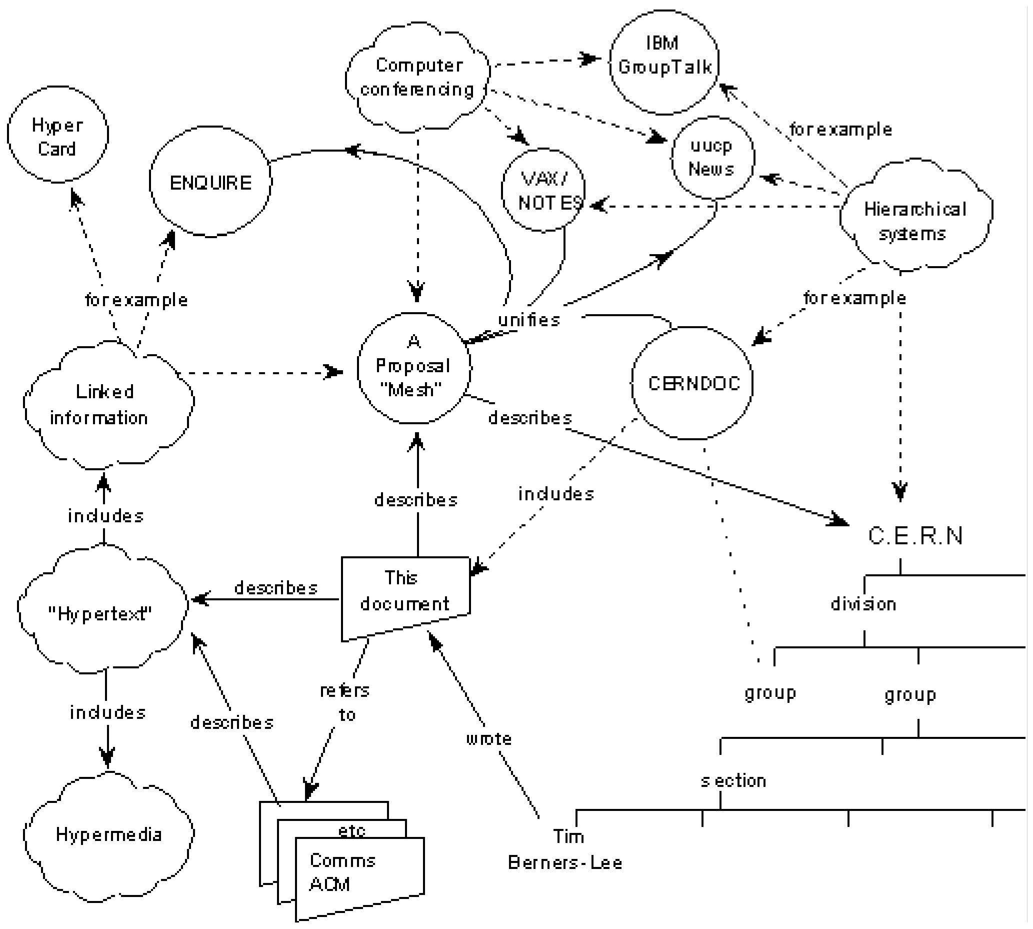


Figure 2 - The inception of the World Wide Web

The evolution

The World Wide Web, originated by an idea of Tim Berners Lee (Figure 2), had an incredible effect in changing the way people is accessing information, and, we could say, the way of life of million or persons. In a few years, we assisted to an evolution (Figure 3) of the web. While at the beginning the attention was mainly focused upon the *man-machine interaction*, now the interest is more and more towards the *machine-to-machine interaction*.

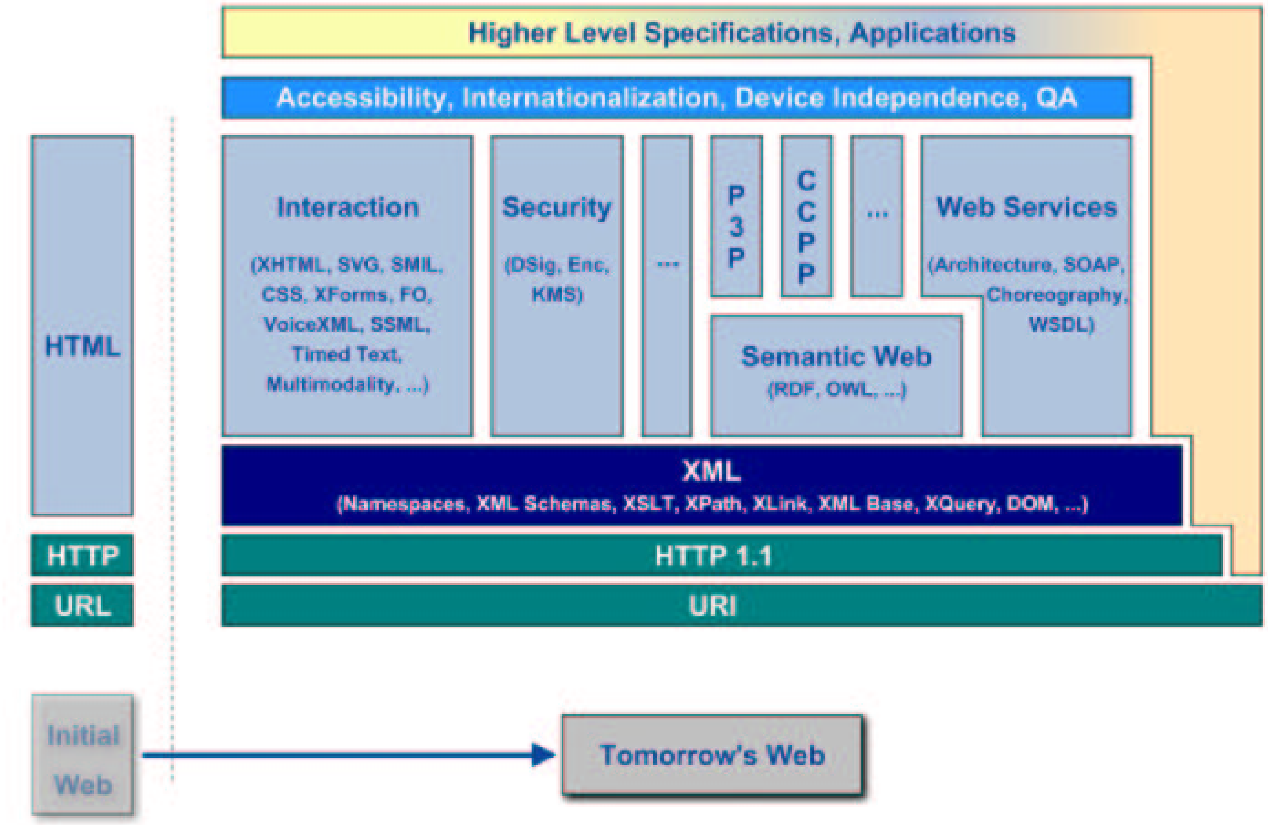

Higher Level Specifications, Applications
Accessibility, Internationalization, Device Independence, QA
Interaction
(XHTML, SVG, SMIL, CSS, XForms, FO, VoiceXML, SSML, Timed Text, Multimodality, ...)
Security
(DSig, Enc, KMS)
...
P3P
CCPP
...
Web Services
(Architecture, SOAP, Choreography, WSDL)
Semantic Web
(RDF, OWL, ...)
XML
(Namespaces, XML Schemas, XSLT, XPath, XLink, XML Base, XQuery, DOM, ...)
HTTP 1.1
URI
HTML
HTTP
URL
Initial Web
Tomorrow's Web

Figure 3 - The evolution of the Web

The need for interoperability

Interoperability means that applications can exchange data and services in a consistent and effective way, facing different hardware and software platforms. In the present world interoperability is a *key success factor*. Advantages of interoperability are well known. Among the other, saving of investments and enlarging the market. Saving of investments is the effect of the ability of facing evolution, and matching solutions offered by other application providers. In addition, the ability to provide interoperable applications/systems can help in enlarging the market (reaching new potential customers, providing better information, enhancing linking to other sites/info).
The basis is to rely on a consistent framework/technology.
However, interoperability is not a merely technological issue. We have to consider differences in cultures and perceptions of concepts, that is, we have to consider not only a *technological interoperability*, but a *semantic interoperability* as well.
In particular, universal access means access to everyone, overcoming differences in culture, language, education, ability, material resources, and physical limitations of users on all continents. This goal immediately leads to the issue of *accessibility* for impaired persons, but also, as the Web should really be a World Wide Web, to the issue of *cultural barriers*. It is well known that expressions, colours, images, can have a different meaning in different cultures. Sometimes, even

the layout of the presentation can suggest a different attitude in readers, a user manual, a book, a newspaper, all have a different look. When designer has clear in his/her mind that a message is made of:

- *Content*: the true content of the message, the originator wants to communicate;
- *Structure*: the way the information is organized (e.g. title, author, body, signature)
- *Presentation*: the way the information is presented to the user (fonts, colours, page layout, etc.)

(s)he can overcome many of these hurdles.
In the following, we will briefly describe some W3C technologies, showing how they can help in handling structure (XML and XMLSchema) and presentation of information (XSL and XSLT). Subsequently, we will discuss how to manage the semantic issues, in the frame of reference of the Semantic Web.

W3C technologies for technological interoperability

Structuring information

XML

Extensible Markup Language (**XML**) was born to overcome HTML limitations in implementing new *data-centric Web applications*. Therefore it was a first step to supply semantics to tags and support web transactions, allowing exchange of information among different databases. Great advantages are the possibility of having different views of the same data, and personalize their presentation by means of appropriate agents. Finally, it must be stressed that XML, independent from platforms and languages, plays a fundamental role towards interoperability.

```
01  <?xml version="1.0"?>
02  <!DOCTYPE order [
03  <!ELEMENT order ( customer, product+ )>
04  <!ATTLIST order id ID #REQUIRED>
05  <!ELEMENT customer EMPTY>
06  <!ATTLIST customer db CDATA #REQUIRED>
07  <!ELEMENT product  ( price )>
08  <!ATTLIST product  db CDATA #REQUIRED>
09  <!ELEMENT price   ( #PCDATA )>
10  ]>
11  <order id="ord001">
12    <customer db="custcode123"/>
13    <product db="prod345">
14      <price>23.45</price>
15    </product>
16  </order>
```

Figure 4 - An XML document (DTD in boldface)

Figure 4 is a simple example of an XML document, describing Order processing. According to XML syntax, every information field, called *element*, must be enclosed between a start and end tag (e.g. `<price>` and `</price>`. An element can also have *attributes*, made by name/value pairs (as, in the example, is id="ord001"). Tags must be correctly nested, we can have empty elements, attribute values must be enclosed in quotes. Syntax is very simple, automatic processing is easy, and source documents remains human readable.
We can have a formal description of the, otherwise implicit document structure, named **DTD** *(Document Type Definition*), that can be included in the document itself, or can be referenced as an external resource. In the example, the expression in line 6 indicates that the attribute db is mandatory. A XML document is "*well formed*" if it matches writing rules, is "*validated*" when its structure is coherent with the referenced DTD.
XML is extensible and flexible, and is the basic technology adopted to model the web. All new languages are described using XML.

XML Schema Definition

DTDs are expressed using their own syntax; therefore they require appropriate editors, parsers, processors. In addition, their extension is not easy, datatype concept is missing. Finally, DTD must support all elements and attributes described in the included namespaces. Schemas play a role similar to DTDs, but offer significant advantages: are written according to XML syntax, include datatypes, inheritance, schema combination rules, offer better namespace support and allow linking of semantic information. Using XMLSchema, designer can specify value constraints, complex types and type hierarchies. As a conclusion, XMLSchemas are much more powerful than DTDs, and designers should carefully consider the opportunity of using them in developing new applications, to save investments and be in line with the evolution of the web.

XML Namespaces

XML is becoming more and more popular, and the possibility of using together different XML applications is increasing. As a consequence, we must be able to distinguish among different namespaces. An **XML namespace** is a *collection of names, identified by a URI reference, which are used in XML documents as element types and attribute names*. XML namespaces differ from the "namespaces" conventionally used in computing disciplines in that the XML version has internal structure and is not, mathematically speaking, a set. To each *nameset* is associated an identifying prefix, and tags are uniquely identified specifying the prefix followed by their “local” name.

```
<!DOCTYPE ordine SYSTEM "OrdDTD">
  < ordine
    xmlns:cli="http://generalstore.it/anagcli"
    xmlns:ordcli="http://generalstore.it /ordcli"
    ordcli:nord="2001Set320">
      <cli:cliente>Il re dei golosi</cli:cliente>
      <rigaord>
        <prodotto>Tartufo d’Alba prima qualità</prodotto>
        <qta>1</qta>
        <umis>Kg</umis>
        - - -
      </rigaord>
  </ ordine >
```

Figure 5 - A document referring a namespace

Presenting information

Transformation and presentation of information

Among the many possibilities offered by XML in implementing innovative applications, it is interesting to emphasize the possibility of implementing architectures with load balancing between client and server, or personalizing the information presentation. The basis is the understanding of the basic principle of *separating content and presentation*. It results in a common reference architecture (Figure 6), where information is extracted from the enterprise database and structured as an XML document; subsequently this XML document is transformed by means of a style transformation into the format that results to be the more appropriate for the end user.
As pointed out in [Holman2000], it is important when we think about styling information to remember that two distinct processes are involved, not just one. First, we must *transform* the information from the organization used when it was created into the organization needed for consumption. Second, when *rendering* we must express, whatever the target medium, the aspects of the appearance of the reorganized information.
Consider the flow of information as a streaming process where information is created upstream and processed or consumed downstream. Upstream, in the early stages, we should be expressing the information abstractly, thus preventing any early binding of concrete or final-form concepts. Midstream, or even downstream, we can exploit the information as long as it remains flexible and abstract. Late binding of the information to a final form can be based on the target use of the final

product; by delaying this binding until late in the process, we preserve the original information for exploitation for other purposes along the way. It is a common but misdirected practice to model information based on how you plan to use it downstream.
In general, there are two distinct styling steps: *transforming* the instance of the XML vocabulary into a new instance according to a vocabulary of rendering semantics; and *formatting* the instance of the rendering vocabulary in the user agent. In order to meet these two distinct processes in a detached (yet related) fashion, the W3C Working Group responsible for the Extensible Stylesheet Language (XSL) split the original drafts of their work into two separate Recommendations: one for transforming information and the other for rendering information.
The *XSL Transformations* (XSLT) 1.0 Recommendation describes a vocabulary recognized by an XSLT processor to transform information from an organization in the source file into a different organization suitable for continued downstream processing.
The Extensible Stylesheet *Language* (XSL) Recommendation describes a vocabulary recognized by a rendering agent to reify abstract expressions of format into a particular medium of presentation.

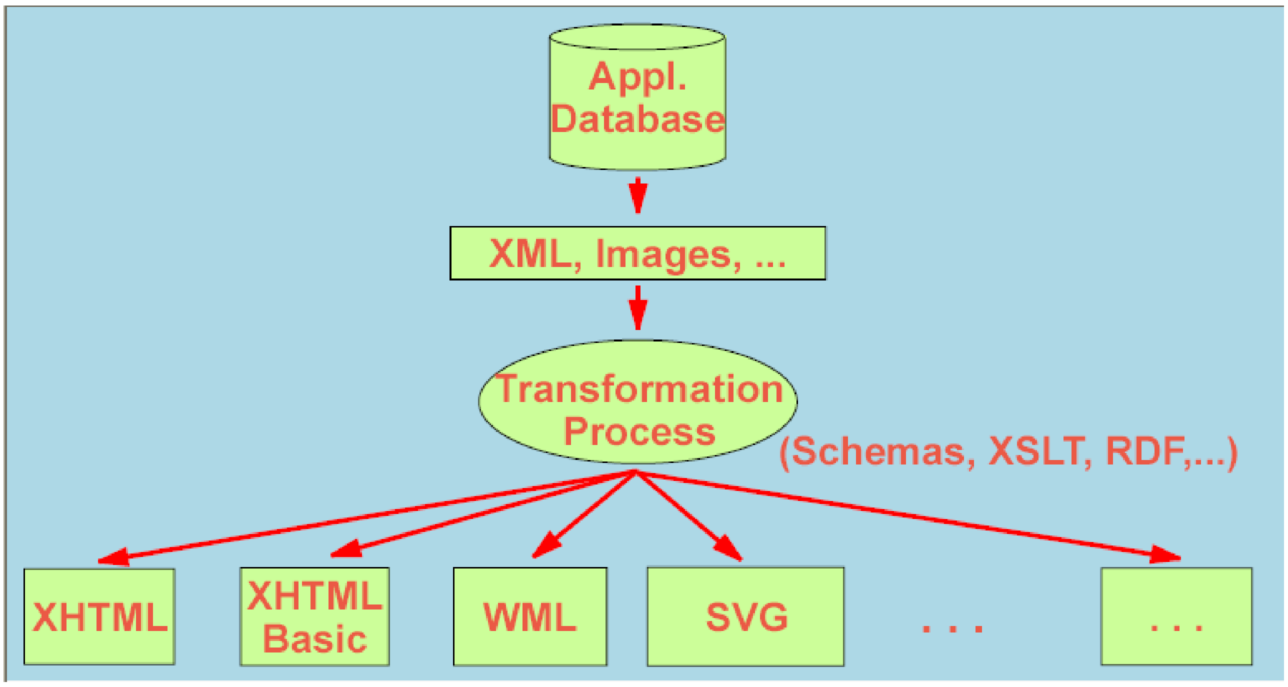


Figure 6 - A reference architecture

It is important to remember that XSLT was designed *primarily for transforming XML vocabularies to the XSL formatting vocabulary*. This doesn't preclude us from using XSLT for other transformation requirements, but it does influence the design of the language and it does constrain some of the functionality from being truly general purpose.
For this reason, the designers *do not* claim XSLT is a general purpose transformation language. However, it is still powerful enough for *most* downstream processing transformation needs, and XSLT stylesheets are often called *XSLT transformation scripts* because they can be used in many areas not at all related to *stylesheet rendering*. Consider an electronic commerce environment where transformation is not used for presentation purposes. In this case, the XSLT processor may transform a source instance, which is based on a particular vocabulary, and *deliver the results to a legacy application* that expects a different vocabulary as input. In other words, we can use XSLT in a non-rendering situation when it doesn't matter what syntax is utilized to represent the content; when only the parsed result of the syntax is material.
An example of the transformation process[1] is shown in Figure 7, Figure 8, Figure 9, Figure 10.

[1] Acknowledgement to Cristian Lucchesi who prepared this material.

```
<?xml version="1.0" ?>
<rubrica>
  <persona>
    <nome>Cristian</nome>
    <cognome>Lucchesi</cognome>
    <telefono>2116</telefono>
    <indirizzo>
       <via>Via Moruzzi,25</via>
       <cap>56100</cap>
       <citta>Pisa</citta>
    </indirizzo>
  </persona>
  <persona>
    <nome>Oreste</nome>
    <cognome>Signore</cognome>
    <telefono>2995</telefono>
  </persona>
</rubrica>
```

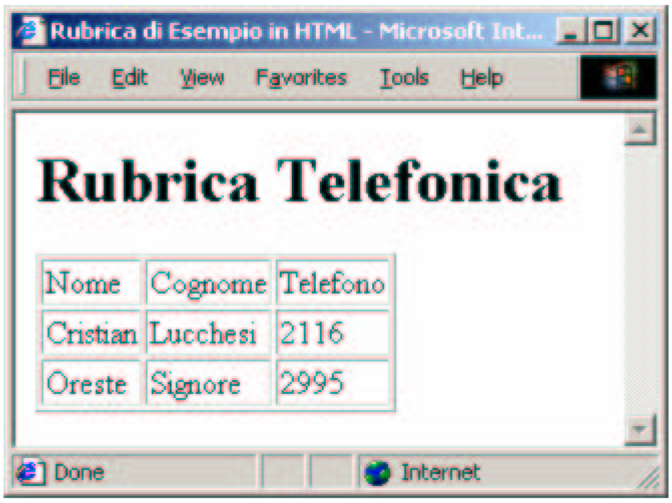
Figure 7 - The XML file for address book (in Italian)

```
<?xml version="1.0" ?>
<xsl:stylesheet
     xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
  <xsl:output doctype-system=
          "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd"
    doctype-public="-//W3C//DTD XHTML 1.0 Transitional//EN"
      indent="yes" media-type="text/html" method="html" />
  <xsl:template match="rubrica">
    <html xml:lang="it" lang="it">
      <head>
        <title>Rubrica di Esempio in HTML</title>
        <meta http-equiv="Content-Type"
        content="text/html; charset=iso-8859-1" />
      </head>
      <body>
        <h1>Rubrica Telefonica</h1>
        <table border="1">
          <tr>
            <td>Nome</td>
            <td>Cognome</td>
            <td>Telefono</td>
          </tr>
          <xsl:apply-templates />
        </table>
      </body>
    </html>
  </xsl:template>
  <xsl:template match="persona">
  <tr>
      <td>
        <xsl:value-of select="nome" />
      </td>
      <td>
        <xsl:value-of select="cognome" />
      </td>
      <td>
        <xsl:value-of select="telefono" />
      </td>
    </tr>
  </xsl:template>
</xsl:stylesheet>
```

Figure 8 - The transformation script

```
<?xml version="1.0"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xml:lang="it" lang="it">
  <head>
    <title>Rubrica di Esempio in HTML</title>
    <meta http-equiv="Content-Type"
          content="text/html; charset=iso-8859-1" />
  </head>
  <body>
    <h1>Rubrica Telefonica</h1>
      <table border="1">
         <tr>
           <td>Nome</td>
           <td>Cognome</td>
           <td>Telefono</td>
        </tr>
        <tr>
          <td>Cristian</td>
          <td>Lucchesi</td>
          <td>2116</td>
        </tr>
        <tr>
          <td>Oreste</td>
          <td>Signore</td>
          <td>2995</td>
        </tr>
      </table>
  </body>
</html>
```

Figure 9 - The address book in XHTML



Rubrica Telefonica

Nome	Cognome	Telefono
Cristian	Lucchesi	2116
Oreste	Signore	2995

Figure 10 - The address book as seen by the end user

All inputs must be well formed XML documents. Therefore we cannot process HTML documents, whose lexical conventions are not XML compliant, while it is possible to process XHTML (*Extensible Hypertext Markup Language*)[2] files.
A three-tiered architecture (Figure 11) can meet technical and business objectives by delivering structured information to web browsers by using XSLT on the *host*, or on the *user agent*, or even on *both*. Considering technical issues first, the server can distribute the processing load to XML/XSLT-aware user agents by delivering a combination of the stylesheet and the source information to be transformed on the recipient's platform. Alternatively, the server can perform the

[2] XHTML files can be generated from HTML files using Tidy (http://www.w3.org/People/Raggett/tidy/), a free tool available on W3C site, which identifies wrong HTML statements, correcting them or issuing warnings.

transformations centrally to accommodate those user agents supporting only HTML, HTML/CSS or WML ([Lee2000]).vocabularies.

o

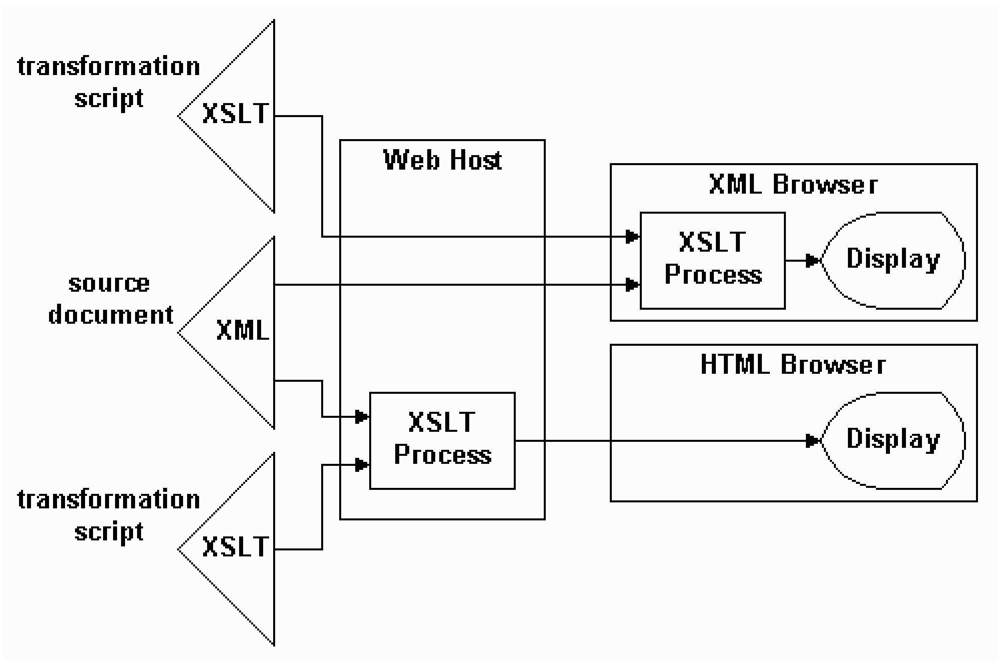


Figure 11 - Transforming information (from [Holman2000])

However, decision about the way of transferring results to the client is not a mere technical issue. XML documents are semantically[3] richer than HTML documents, but it can happen that system designer prefer do not make available to everyone the full information contained in an XML marked up document. A possibility is to modify the document content according to the user's characteristics. Another one is designing two XSLT processes, the first one to transform information markup into a generic distribution format, the second one to be executed on the client agent side, to personalise the presentation. Anyway, there are two different way of thinking. The first one supports the idea of having a *"semantic firewall*" between enterprise data and user, who should receive pure HTML data, without any semantics. The second one is in favour of moving towards a *semantic web* where semantically rich transmitted information should be processed by an intelligent user agent, which will conform to appropriate rules.

Multimedia

The **S**ynchronized **M**ultimedia **I**ntegration **L**anguage (SMIL, pronounced "smile") enables simple authoring of interactive audiovisual presentations. SMIL is typically used for "rich media"/multimedia presentations which integrate streaming audio and video with images, text or any other media type. SMIL is an easy-to-learn HTML-like language, and many SMIL presentations are written using a simple text-editor.
SMIL, a W3C Recommendation, is an XML application. In essence designers can specify what and when present, synchronizing text presentation, sound, images. The basic idea is to use an URI to assign a name to every component, independently they are text, image, sound, video, and program their execution as in sequence or in parallel. It is worthwhile to note that all the elements are supposed to reside on a web server, and are accessible by their URI. Without going in more detail, we will just use an example (Figure 12) to show the usage of `par` (parallel) and `excl` (exclusive) tags. The sample page contains two imagines as buttons. Clicking on one of them, the user activates one, and only one, of them. If the user selects "Story", and afterwards "Weather", playing of "Story" is stopped, and a map ("Weather.jpg") is showed. The `excl` element makes possible the selection

[3] Really, semantic richness of XML tagging is not different from the column names in a relational database.

of one option at a time. The `par` element makes possible presentation of caption while video is played, and playing of weather forecast while the weather conditions map is showed.

```
   <par>
      <a href="#Story"> <img src="button1.jpg"/> </a>
      <a href="#Weather"> <img src="button2.jpg"/></a>
       <excl>
           <par id="Story " begin="0s">
             <video src="video1.mpg"/>
             <text src="caption.html"/>
           </par>

           <par id="Weather">
             <img src="weather.jpg"/>
             <audio src="weatherForecast.mp3"/>
           </par>
       </excl>
   </par>
```

Figure 12 - A simple SMIL application

```
   <mrow>
     <mi>x</mi> <mo>=</mo>
     <mfrac>
       <mrow>
         <mrow><mo>-</mo><mi>b</mi></mrow>
         <mo>&PlusMinus;</mo>
         <msqrt>
           <mrow>
             <msup><mi>b</mi><mn>2</mn></msup>
             <mo>-</mo>
             <mrow>
               <mn>4</mn><mo>&InvisibleTimes;</mo>
               <mi>a</mi><mo>&InvisibleTimes;</mo>
               <mi>c</mi>
             </mrow>
           </mrow>
         </msqrt>
       </mrow>
       <mrow>
         <mn>2</mn><mo>&InvisibleTimes;</mo>
         <mi>a</mi>
       </mrow>
     </mfrac>
   </mrow>
```

Figure 13 – The quadratic formula expressed in MathML

Mathematics and graphics

Mathematics and graphics are everywhere on the Web. Superscripts and subscripts make quite easy to insert mathematical formulae in HTML pages (as an example, a_i could be expressed as: `a<sub>i</sub>`). More complex formulae (yet the formula for quadratic equation) require some tricks. The most common approach is to have the formula as an image, pasting it in the text. Apart the well known problems of exact positioning, the formula is seen as a single element: no part of it can be separately identified; a visually impaired person can't have access to it. Finally, putting on the web existing material would be expensive and time consuming.
MathML ([MathML]) is a markup language that can be used to write mathematical complex formulae, using XML syntax. Single elements in the formula are individually searchable, formulae can be mixed with other markups, and a voice browser could render the content to a visually impaired user.
Figure 13 shows how we can express the quadratic formula, using MathML.

Another component for information presentation is graphics. SVG (**S**calable **V**ector **G**raphics) is the W3C Recommendation for vector graphics. SVG is a language for describing two-dimensional

graphics in XML. SVG allows for three types of graphic objects: vector graphic shapes (e.g., paths consisting of straight lines and curves), images and text. Graphical objects can be grouped, styled, transformed and composited into previously rendered objects. Text can be in any XML namespace suitable to the application, which enhances searchability and accessibility of the SVG graphics. Graphical SVG objects are therefore scalable, have components that can be individually identified and enriched of semantic descriptions (metadata), and be origin or target of (possibly semantically enriched) links. The feature set includes nested transformations, clipping paths, alpha masks, filter effects, template objects and extensibility.
SVG drawings can be dynamic and interactive. The Document Object Model (DOM) for SVG, which includes the full XML DOM, allows for straightforward and efficient vector graphics animation via scripting. A rich set of event handlers such as `onmouseover` and `onclick` can be assigned to any SVG graphical object. Because of its compatibility and leveraging of other Web standards, features like scripting can be done on SVG elements and other XML elements from different namespaces simultaneously within the same Web page.

Design for all

Web accessibility[4] is important for several reasons:

- use of the Web is spreading rapidly into all areas of society;
- there are barriers on the Web for many types of disabilities;
- millions of people have disabilities that affect access to the Web;
- Web accessibility has carry-over benefits for other users;
- some Web sites *are required* to be accessible.

In addition, we can't forget that the Web is the fastest-adopted technology in history, and for people with disabilities, it's sometimes a "mixed blessing": In fact it is displacing traditional sources of information and interaction, like schools, libraries, print materials, discourse of the workplace. Some of the traditional resources were accessible; some not.
The Web is becoming a key, but sometimes inaccessible, resource for *information gathering* (news, information, commerce, entertainment), *education* (classroom education, distance learning), *employment* (job searching, and workplace interaction), *civic participation* (laws, voting, government information, services). An accessible Web will mean unprecedented access to information for people with disabilities. Further, *Web accessibility* is a cross-disability issue, as the Web can present barriers to people with different kinds of disabilities:

- *visual* disabilities (unlabeled graphics, undescribed video, poorly marked-up tables or frames, lack of keyboard support or screen reader compatibility);
- *hearing* disabilities (lack of captioning for audio, proliferation of text without visual signposts);
- *physical* disabilities (lack of keyboard or single-switch support for menu commands);
- *cognitive or neurological* disabilities (lack of consistent navigation structure, overly complex presentation or language, lack of illustrative non-text materials, flickering or strobing designs on pages).

However, Web accessibility is not only an unmissable target for helping impaired persons, it is also a *marketplace* issue, and few organizations can afford to deliberately miss this market sector. In fact, 10% to 20% of the population in most countries has disabilities, and average age of population in many countries is increasing. Even if not all disabilities affect access to the Web (for example difficulty walking, heart condition, etc., don't, while vision, hearing, dexterity, short-term memory problems do), we must consider that aging sometimes results in combinations of accessibility issues (like vision and hearing changes, dexterity).
Moreover, accessibility contributes to better design for other users, and therefore to Universal Design, as it helps when users are in special conditions, like *low bandwidth* (images are slow to

[4] "*The power of the Web is in its universality. Access by everyone regardless of disability is an essential aspect.*" -- Tim Berners-Lee, W3C Director and inventor of the World Wide Web

download), *noisy environments* (difficult to hear the audio), *screen-glare* (difficult to see the screen), *driving* (eyes and hands are "busy"), different learning styles, low literacy levels, second-language access, and so on.
Due to its importance, a number of governments require Web accessibility for certain kinds of sites, often for government Web sites first, sometimes other sites, to implement anti-discrimination policies, or policies that directly address Web accessibility.
The Web Accessibility Initiative (WAI) works across all the other domains of the W3C and coordinates with other W3C working groups to ensure that Web technologies support accessibility. Many specifications ([HTML-AF], [CSS-AF], [SMIL-AF], [MathML]) already include support for accessibility.
In addition, WAI is working on accessibility issues in many current areas of W3C technology development. Guidelines play a critical role in making the Web accessible, by explaining how to use Web technologies to create accessible Web sites, browsers, or authoring tools. WAI has three different guidelines to address these different needs:

- **Web Content Accessibility Guidelines** 1.0 ([WCAG]) which explain to authors how to create accessible. It identifies three *priority levels* and three *conformance levels*.
- **Authoring Tool Accessibility Guidelines** 1.0 ([ATAG]) which explains to developers how to design authoring tools that are accessible to authors with disabilities, and that produce accessible Web content, conformant to WCAG 1.0.
- ***User Agent Accessibility Guidelines*** *1.0*, ([UAAG]) which explain what the software developers can do to improve the accessibility of mainstream browsers and multimedia players so that people with hearing, cognitive, physical, and visual disabilities will have improved access to the Web.

Towards semantic interoperability

Metadata

Navigating the web, we follow links. The thing which we get when we de-reference a URI[5], is formally called a *resource*. Sometimes it is referred to as a *document* to emphasize that it is human readable, or as *object* to emphasize that it is something which is more machine readable in nature or has hidden state.
One of the characteristics of the World Wide Web is that resources, when you retrieve them, do not stand simply by themselves without explanation, but there is information about the resource. Information about information is generally known as *Metadata*. We can therefore give the following definition: *Metadata is machine understandable information about web resources or other things*.
The phrase "*machine understandable*" is the key. Metadata is information which *software agents* can use in order to make life easier for us, ensure we obey our principles, make appropriate and efficient use of resources, check that we can trust what we are doing, and make everything work more smoothly and rapidly. For an example of metadata, when an object is retrieved using the HTTP protocol, the protocol allows information about its date, its expiry date, its owner, and other arbitrary information to be sent by the server. The present World Wide Web is therefore a world of information and some of that information is information about information.
To better understand the metadata concept, we must keep clear in our minds that *metadata is data*, and this has some consequences:

- *metadata can be stored regarded as data*, it can be stored in a resource. So, one resource may contain information about itself or about another resource. Metadata about one

[5] **URI** (*Uniform Resource Identifier*): the generic set of all names/addresses that are short strings that refer to resources
URL (*Uniform Resource Locator*): an informal term (no longer used in technical specifications) associated with popular URI schemes: http, ftp, mailto, etc
For more details, refer to [Naming])

document can occur within the document, or within a separate document, or it may be transferred accompanying the document.

- *Metadata can describe metadata*. That is, metadata itself may have attributes such as ownership and an expiry date, and so there is meta-metadata but we don't distinguish many levels, we just say that metadata is data and that from that it follows that it can have other data about itself. This gives the Web a certain consistency.

Metadata has well defined semantics and structure.

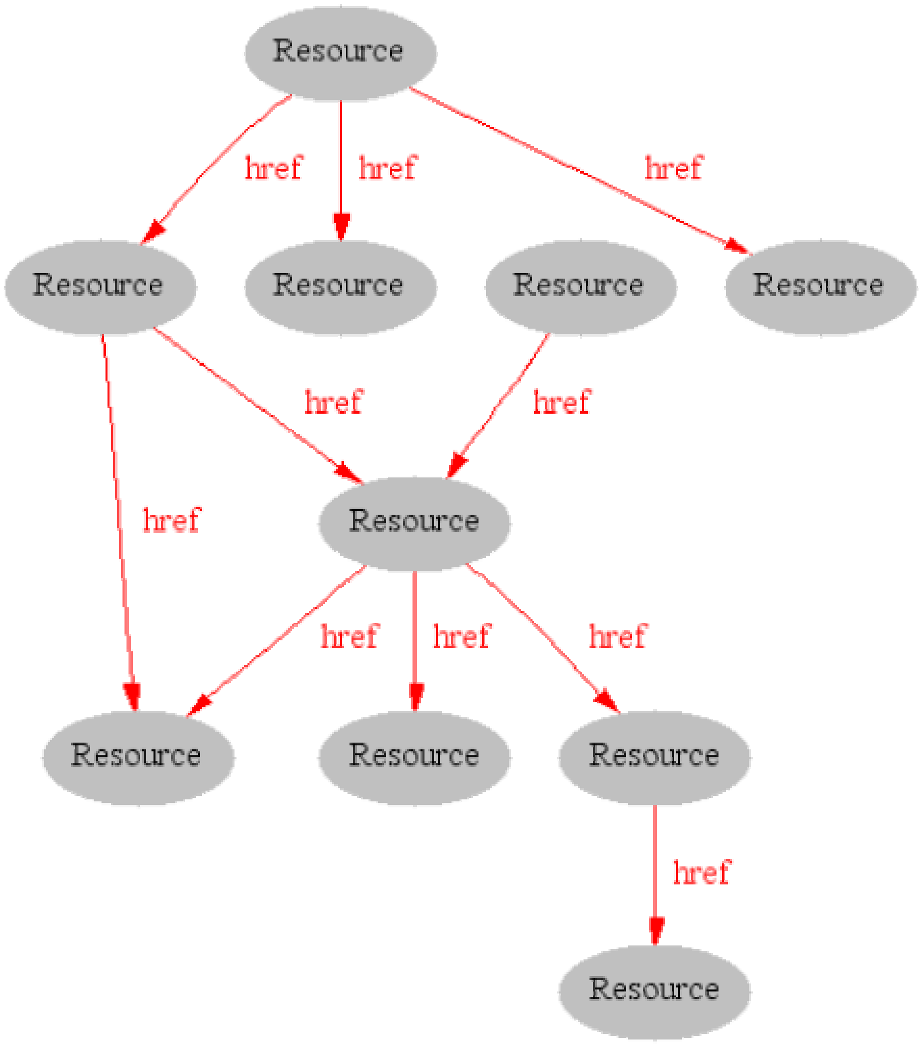


Figure 14 - The Web prior RDF

Resource Description Framework

Metadata offers a viable solution to the ambitious target of automating the Web, while maintaining consistency with its original architecture, where all information was *machine-readable*, but not *machine-understandable*. However, effective use of metadata requires establish appropriate conventions for s*emantics*, *syntax* and *structure*.

Resource Description Framework (**RDF**) is a foundation for processing metadata; it provides interoperability between applications that exchange machine-understandable information on the Web. RDF emphasizes facilities to enable automated processing of Web resources. RDF can be used in a variety of application areas; for example: in *resource discovery* to provide better search engine capabilities, in *cataloguing* for describing the content and content relationships available at a particular Web site, page, or digital library, by *intelligent software agents* to facilitate knowledge sharing and exchange, in *content rating*, in describing *collections of pages* that represent a single logical "document", for describing *intellectual property rights* of Web pages, and for expressing the

privacy preferences of a user as well as the *privacy policies* of a Web site. RDF with *digital signatures* will be key to building the "Web of Trust" for electronic commerce, collaboration, and other applications.
RDF is not describing semantics, but provides a common basis to express it, so allowing defining the XML tags semantics. RDF is made by two components ([RDFMSS], [RDFSS]):

- **RDF Model and Syntax**: defines the RDF *data model*;
- **RDF Schema**: allows the definition of specific *metadata vocabularies*.

Role and effects of the RDF are simply sketched in Figure 14 and Figure 15.
An overview of the model follows.

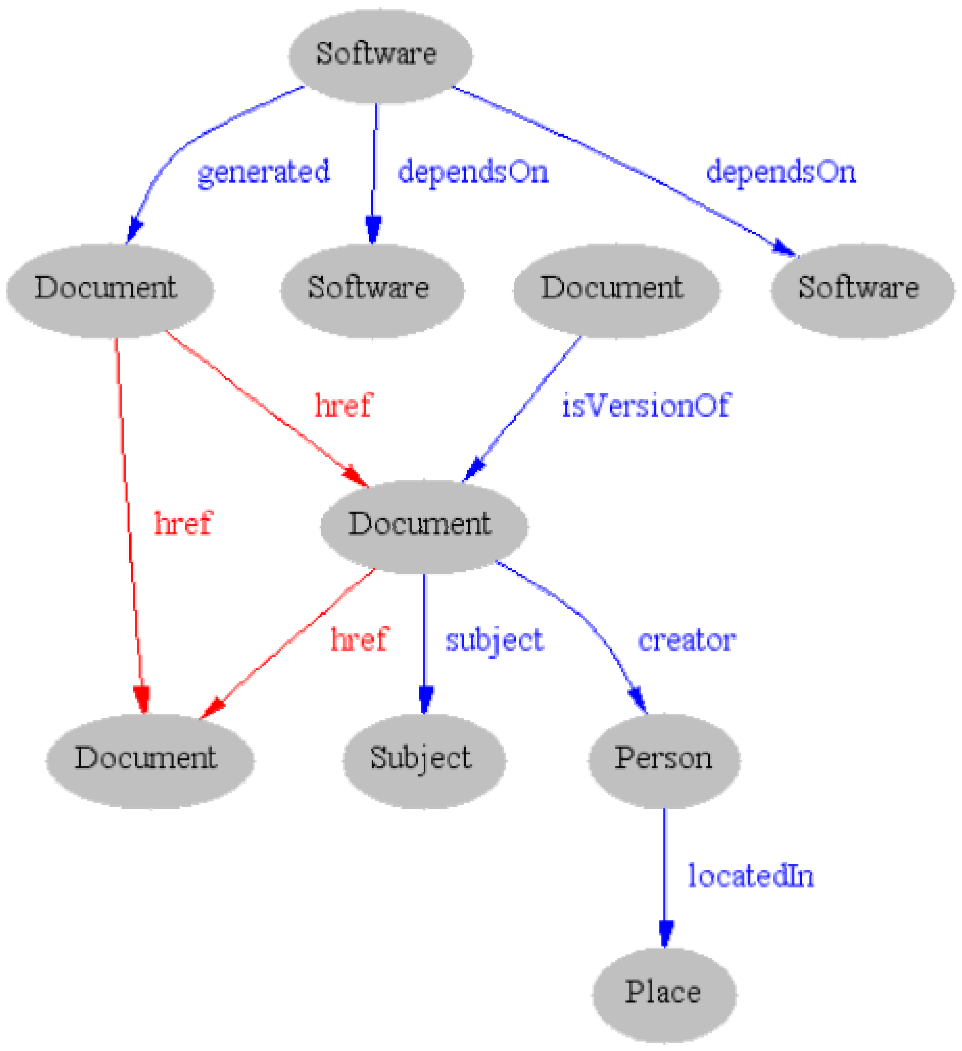


Figure 15 - The Web after RDF

RDF Data Model

The foundation of RDF is a model for representing named properties and property values. The RDF model draws on well-established principles from various data representation communities. RDF properties may be thought of as attributes of resources and in this sense correspond to traditional attribute-value pairs. RDF properties also represent relationships between resources and an RDF model can therefore resemble an entity-relationship diagram. (More precisely, RDF Schemas, which are themselves instances of RDF data models, are ER diagrams.) In object-oriented design terminology, resources correspond to objects and properties correspond to instance variables.

The RDF data model is a syntax-neutral way of representing RDF expressions. The basic data model consists of three object types:

Resources All things being described by RDF expressions are called *resources*. A resource may be an entire Web page; or be a part of it, (e.g. a specific HTML or XML element within the document source). A resource may also be a whole collection of pages (e.g. an entire Web site) or be an object that is not directly accessible via the Web (e.g. a book, a painting, etc.) Resources are always named by URIs plus optional anchor ids. Anything can have a URI; the extensibility of URIs allows the introduction of identifiers for any entity imaginable.

Properties A *property* is a specific aspect, characteristic, attribute, or relation used to describe a resource. Each property has a specific meaning, defines its permitted values, the types of resources it can describe, and its relationship with other properties. Each property is identified by a *name*, and takes some *values*.

Statements A specific resource together with a named property plus the value of that property for that resource is an RDF *statement*. These three individual parts of a statement are called, respectively, the *subject*, the *predicate*, and the *object*. The object of a statement (i.e., the property value) can be another resource or it can be a literal; i.e., a resource (specified by a URI) or a simple string or other primitive datatype defined by XML

A set of properties referring the same resource is called *description*.
We can diagram an RDF statement pictorially using *directed labelled graphs* (also called "nodes and arcs diagrams"). In these diagrams, the nodes (drawn as ovals) represent resources and arcs represent named properties. Nodes that represent string literals will be drawn as rectangles. The sentence:

> *The individual referred to by employee id 85740 is named Ora Lassila and has the email address lassila@w3.org. The resource http://www.w3.org/Home/Lassila was created by this individual.*

would thus be diagrammed as in Figure 16

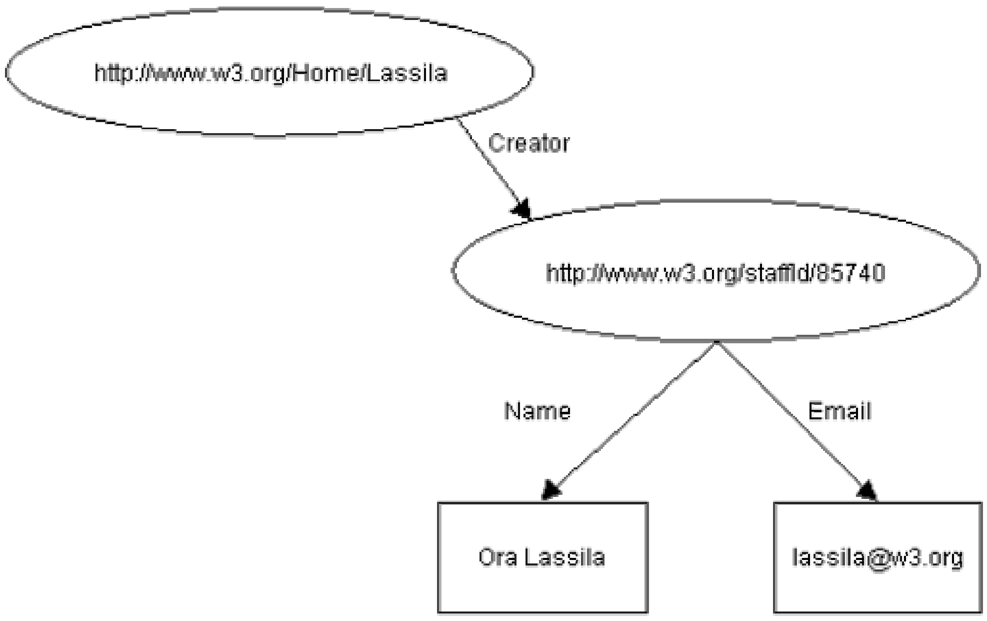

http://www.w3.org/Home/Lassila
Creator
http://www.w3.org/staffId/85740
Name
Email
Ora Lassila
lassila@w3.org

Figure 16 - A graphical representation of a complex RDF statement

Namespaces: a key for peer to peer architecture

RDF supports the use of conventions that will facilitate *modular interoperability among separate metadata element sets*. These conventions include standard mechanisms for representing semantics that are grounded in the simple, yet powerful, data model discussed before. RDF additionally provides a means for publishing both human-readable and machine-processable vocabularies. Vocabularies are the set of properties, or metadata elements, defined by resource description communities. The ability to standardize the declaration of vocabularies is anticipated to encourage the reuse and extension of semantics among disparate information communities. RDF uniquely identifies properties using namespaces [XMLns]), that provide a way to unambiguously identify semantics and rules governing properties usage specifying the authority managing the vocabularies.
A well known example is the Dublin Core Initiative ([DC]) which defines, for example, the "*Subject and Keywords*" field in the following way:

```
Name:      Subject and Keywords
  Identifier:Subject
  Definition:The topic of the content of the resource.
  Comment: Typically, a Subject will be expressed as keywords, key phrases or
           classification codes that describe a topic of the resource.
           Recommended best practice is to select a value from a controlled vocabulary or
           formal classification scheme.
```

We can use an XML namespace to unambiguously identify the Dublin Core vocabulary just pointing to the Dublin Core resources defining its semantics. Description of a web site using Dublin Core properties could be as follows:

```
<rdf:RDF
  xmlns:rdf=http://www.w3.org/1999/02/22-rdf-syntax-ns#
  xmlns:dc="http://purl.org/metadata/dublin_core#"
  xmlns:mydc="http://www.w3c.it/metadata/DCaddendum#">
  <rdf:Description about="http://www.dlib.org">
    <dc:Title>
        D-Lib Program - Research in Digital Libraries
    </dc:Title>
    <dc:Description>
        The D-Lib program supports the community of people
        with research interests in digital libraries and
        electronic publishing
   .</dc:Description>
    <dc:Publisher>
        Corporation For National Research Initiatives
    </dc:Publisher>
    <dc:Date>1995-01-07</dc:Date>
    <dc:Subject>
      <rdf:Bag>
        <rdf:li>Research; statistical methods</rdf:li>
        <rdf:li>Education, research, related topics</rdf:li>
        <rdf:li>Library use Studies</rdf:li>
      </rdf:Bag>
    </dc:Subject>
    <dc:Type>World Wide Web Home Page</dc:Type>
    <dc:Format>text/html</dc:Format>
    <dc:Language>en</dc:Language>
    <mydc:Rating>
       Well known and often referenced site
    </mydc:Rating>
    <mydc:Originality>High</mydc:Originality>
  </rdf:Description>
</rdf:RDF>
```

This example is similar to the one in [RDFMSS], where three *namespaces*, referenced by the **`rdf`**, **`dc`** and **`mydc`**, prefixes allow usage of properties defined there. In particular, mydc namespace allows addition of new properties to the Dublin Core defined ones.

The "Semantic Web Stack"

The real challenge of next years is the *Semantic Web*. According to the vision of Tim Berners-Lee, the Web has a layered architecture (Figure 17), which will be developed in several years.

To understand the framework, we recall that the Web must be seen as a Universal Information Space, navigable, with a mapping from *URI (Uniform Resource Identifier*) to resources. In this framework, the adjective semantic means *"machine processable"*. The Semantic Web, much like XML, is a declarative environment, where we specify the meaning of data.
As clearly explained in [TBL2001], for the semantic web to function, it is needed that computers have access to *structured collection of information* and set of *inference rules* that they can use to conduct automated reasoning. The challenge of the semantic web, therefore, is to provide a language that expresses both *data* and *rules* for reasoning about data and that allows rules from any existing knowledge-representation system to be *exported* onto the Web.

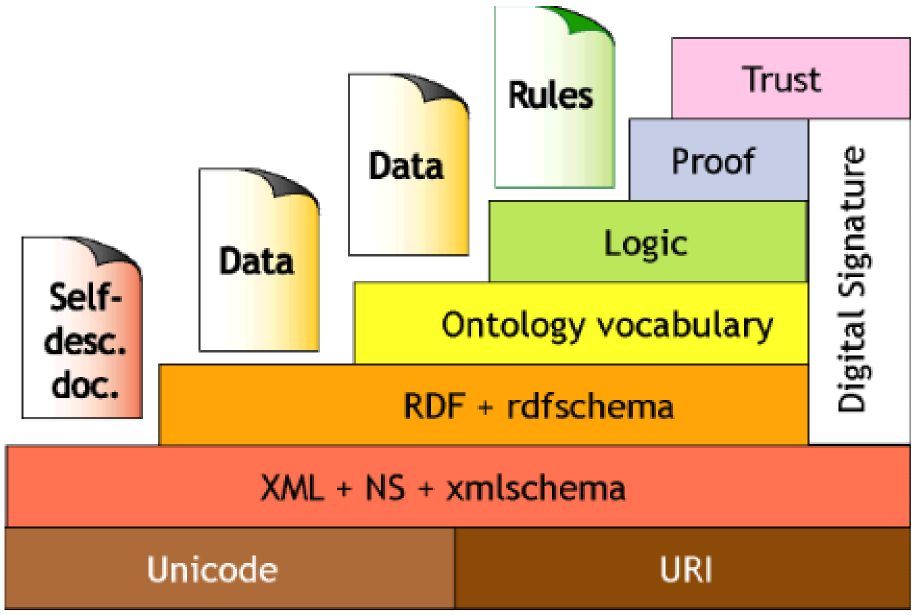


Figure 17 -The Semantic Web stack (http://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html)

Going in more detail, Figure 17 clearly shows the importance of XML, which allows users to add arbitrary structure to their documents, and RDF, which can be used to express the meaning, asserting that particular things have properties (like *author-of*). A third component is the *Ontology Vocabulary*, intended as a source that formally defines the relations among terms. Ontologies can have a significant effect in enhancing functioning of the Web (looking for concepts, relating the information on a page to the associated knowledge structures, etc.). Fine grained *digital signature* allows signing different components, like ontologies, inferences, data, the user can trust in.
Scientific community ([SemWeb]) is investing a lot of energies in the Semantic Web research area.

Web Services: a case for interoperability

Web Services Description

Web services are modelled as interfaces between applications on the World Wide Web. In order to make use of a Web service, we need to know what information is expected and in what form (see Figure 18). Therefore, the interface needs to be explained for use by remote parties. The structure of the messages accepted and/or generated needs to be described.
The W3C Web Services Description Working Group is chartered to design the following components of the interface:

- the *message*: a definition for the types and structures of the data being exchanged.
- the *message exchange patterns*: the descriptions of the sequence of operations supported by a Web service.

- the protocol *binding*: a mechanism for binding a protocol used by a Web service, independently of its message exchange patterns and its messages.

Furthermore, the following three general requirements must be met by the work produced by this Working Group:

- The language developed by the Working Group must not preclude any programming model, nor assume any particular mode of communication between peers.
- Focus must be put on *simplicity*, *modularity* and *decentralization*.
- The language proposed must support the kind of extensibility actually seen on the Web: disparity of document formats and protocols used to communicate, mixing of XML vocabularies using XML namespaces, development of solutions in a distributed environment without a central authority, etc. In particular, it must support distributed extensibility.

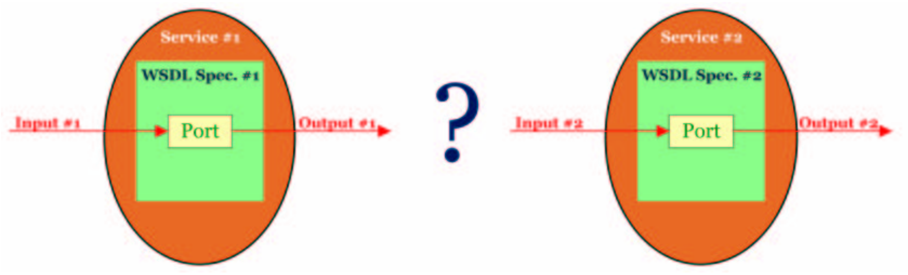


Figure 18 - Communication between Web Services

The W3C Web Services Description Working Group will define a description language expressed in XML. The description language must describe the messages available via the interface, accepted and generated by the Web service. The language must also describe the error messages generated, if any.

The data exchanged is usually typed and structured. This increases interoperability by having applications agreeing on semantics and also provides some level of error detection.

The description language designed will be used both by applications in order to automatically communicate between each other as well as by programmers developing Web services themselves. The language should therefore provide, in addition to the raw XML definition of the interface, human-readable comment capabilities to allow both applications and developers to make use of them.

The information exchanged to and from a Web service can be carried in a large number of different ways. The action of carrying some XML-based communication in an underlying protocol is called, in the XML Protocol jargon, a binding.

The description language defined should therefore describe how to reach the Web Service in a form which is orthogonal to its message exchange patterns and its messages.

It is expected that in the near-term future, Web services will be accessed largely through SOAP Version 1.2 carried over HTTP/1.1, or by means of simple HTTP/1.1 GET and POST requests.

The Semantic Web aims at transforming the Web into a completely machine-processable information space. Web services are aiming at enabling automated distributed computing on the World Wide Web. The work of the Web Services Description Working Group is essentially the provision of an *ontology* for Web services.

One of the tools used by Semantic Web-enabled technologies is RDF. The Working Group will provide a mapping to RDF so that the information described can be easily merged with that of other applications.

SOAP 1.2

SOAP is a fundamental cornerstone for the Web. In a nutshell, SOAP 1.2 ([SOAP12]) provides the definition of the XML-based information which can be used for exchanging structured and typed information between peers in a decentralized, distributed environment. SOAP is fundamentally a

stateless, one-way message exchange paradigm, but applications can create more complex interaction patterns (e.g., request/response, request/multiple responses, etc.) by combining such one-way exchanges with features provided by an underlying protocol and/or application-specific information.
Figure 19 shows the essence of a SOAP message: it is made by an *Envelope* containing two elements: *Header* (optional) and *Body*.

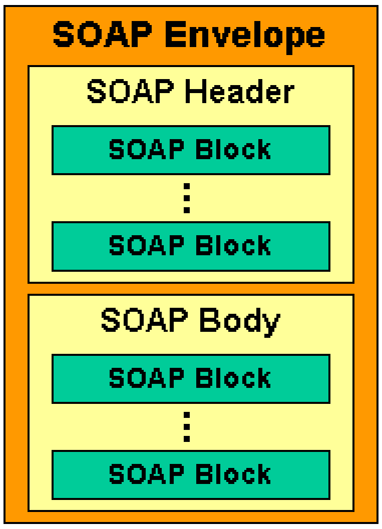


Figure 19 - How a SOAP message is structured

A *SOAP header* element is optional. Its immediate child elements are called *header blocks*, and represent a logical grouping of data which can individually be targeted at SOAP nodes that might be encountered in the path of a message from a sender to an ultimate receiver.SOAP headers have been designed in anticipation of various uses for SOAP, many of which will involve the participation of other SOAP processing nodes - called *SOAP intermediaries* - along a message's path from *an initial SOAP sender* to an *ultimate SOAP receiver*. This may be allows SOAP intermediaries to provide value-added services.
The *SOAP body* is the mandatory element within the SOAP Envelope which implies that this is where the main end-to-end information conveyed in a SOAP message must be carried.

Conclusion

In the evolution from the Web of today towards the Semantic Web, importance of interoperability, both technological and semantic, is getting higher importance. The family of XML technologies is playing its role at different architecture levels.
At the *markup* level, XML contributes to the *interoperability within applications*.
At the *data* level, RDF is the key for *cross applications interoperability*.
Finally, at the *ontology* level, languages like OWL can help in achieving the target of a *web of meaning*.
In this paper we showed how XML and XML applications can be used to present and personalize rich content information, the importance of metadata and the RDF model, the role of these technologies in Web Services. It emerges how conformance to the W3C technologies helps to implement effective, portable, interoperable applications, with a significant saving of investments maintaining coherence with web design principles and its evolution.

Acknowledgement

Many thanks are deserved to all who kindly helped me in preparing this paper, with useful suggestions or illuminating discussions. A special acknowledgement goes to the W3C Team, which

maintains up to date documentation on the official site: some content of this paper comes directly from the W3C Web site pages.

References

[ALTifier] *ALTifier Web Accessibility Enhancement Tool*, http://www.vorburger.ch/projects/alt/

[ATAG] *Authoring Tool Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-AUTOOLS/

[ATAG] *Authoring Tool Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-AUTOOLS/

[Bobby] CAST Bobby, http://www.cast.org/bobby/

[CSS-AF] *CSS2 accessibility features*, http://www.w3.org/TR/CSS-access

[DC] The Dublin Core Home Page, URL: http://dublincore.org/

[DOM] *Document Object Model (DOM)*, http://www.w3.org/DOM/

[Holman2000] Holman G. Ken: *What is XSLT? (I)* http://www.xml.com/lpt/a/2000/08/holman/s1.html

[HTML-AF] *HTML 4.0 accessibility features*, http://www.w3.org/WAI/References/HTML4-access

[HTML-AF] WAI Resource: HTML 4.0 Accessibility Improvements, http://www.w3.org/WAI/References/HTML4-access

[IRDF] Introduction to RDF Metadata, W3C NOTE 1997-11-13, Ora Lassila, URL:http://www.w3.org/TR/NOTE-rdf-simple-intro

[Lee2000] Lee Wei Meng: *Transforming XML into WML*, http://www.wirelessdevnet.com/channels/wap/training/xslt_wml.html

[LitMach1993] Nelson Theodor Holm: *Literary Machines 93.1*, http://www.sfc.keio.ac.jp/~ted/TN/PUBS/LM/LMpage.html

[MathML] W3C's Math Home Page, http://www.w3.org/Math/

[Miller1998] Miller E.: An Introduction to the Resource Description Framework, D-Lib Magazine, May 1998, http://www.dlib.org/dlib/may98/miller/05miller.html

[Naming] Naming and Addressing: URIs, URLs, ..., http://www.w3.org/Addressing/

[NelsonTH] http://www.sfc.keio.ac.jp/~ted/

[Nielsen2000] Nielsen J.: *Designing Web Usability: The Practice of Simplicity*, New Riders Publishing, Indianapolis, 2000, ISBN 1-56205-810-X.

[RDFMSS] O.Lassila, R.Swick: *Resource Description Framework (RDF) Model and Syntax Specification*,W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax

[RDFMSS] O.Lassila, R.Swick: *Resource Description Framework (RDF) Model and Syntax Specification*,W3C Recommendation 22 February 1999, http://www.w3.org/TR/REC-rdf-syntax

[RDFSS] *Resource Description Framework (RDF) Schema Specification*, W3C Recommendation 03 March 1999, http://www.w3.org/TR/1999/PR-rdf-schema-19990303

[RDFSS] *Resource Description Framework (RDF) Schema Specification*, W3C Recommendation 03 March 1999, http://www.w3.org/TR/1999/PR-rdf-schema-19990303

[SemWeb] http://www.semanticweb.org/

[Signore2001] Signore O.: *Il ruolo centrale di XML nell' evoluzione del Web*, XML Day Milan, Conference proceedings, Milan, September 21 (find this and othe similar papers at: http://www.w3c.it/papers/)

[SMIL-AF] *SMIL accessibility features*, http://www.w3.org/WAI/EO/SMIL-access

[TBL1997] Tim Berners-Lee: *Metadata architecture*, (1997), http://www.w3.org/DesignIssues/Metadata.html

[TBL1998] Tim Berners-Lee: *Semantic Web Road Map*, (1998), http://www.w3.org/DesignIssues/Semantic.html

[TBL1999] Tim Berners-Lee: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*, HarperSanFrancisco (1999), ISBN 0-06-251587-X

[TBL1999] Tim Berners-Lee: *Weaving the Web: The Original Design and Ultimate Destiny of the World Wide Web by Its Inventor*, HarperSanFrancisco (1999), ISBN 0-06-251587-X

[TBL2001] Berners-Lee T., Hendler J., Lassila O.: *The Semantic Web*, Scientic American, May 2001, http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

[TBL2001] Berners-Lee T., Hendler J., Lassila O.: *The Semantic Web*, Scientific American, May 2001, http://www.scientificamerican.com/2001/0501issue/0501berners-lee.html

[TUUAG] *Techniques for User Agent Accessibility Guidelines*, http://www.w3.org/WAI/UA/WAI-USERAGENT-TECHS/

[UAAG] *User Agent Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-USERAGENT/

[UAAG] *User Agent Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-USERAGENT/

[UAAG] *User Agent Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-USERAGENT/

[W3C] *World Wide Web Consortium Home Page*, http://www.w3.org

[W3C] *World Wide Web Consortium Home Page*, http://www.w3.org

[W3CML] World Wide Web Consortium (W3C) Members - http://www.w3.org/Consortium/Member/List

[WAI-AERT] *Techniques For Accessibility Evaluation And Repair Tools*, http://www.w3.org/TR/AERT

[WAI-ER] Evaluation and Repair Tools Working Group, http://www.w3.org/WAI/ER/

[WAI-Tablin] *WAI Table Linearizer*, http://www.w3.org/WAI/ER/WG/tablin/

[WAI-Tools] Evaluation, Repair, and Transformation Tools for Web Content Accessibility, http://www.w3.org/WAI/ER/existingtools.html

[WA-Policies] *Policies Relating to Web Accessibility*, http://www.w3.org/WAI/Policy/

[WCAG] *Web Content Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-WEBCONTENT/

[WCAG] *Web Content Accessibility Guidelines 1.0*, http://www.w3.org/TR/WAI-WEBCONTENT/

[XLL] *XML Linking Language (XLink)*, W3C Working Draft 21 February 2000 http://www.w3.org/TR/xlink/

[XML] *Extensible Markup Language (XML)* , http://www.w3.org/XML

[XML] *Extensible Markup Language (XML)* , http://www.w3.org/XML/

[XML1.0] *Extensible Markup Language (XML) 1.0 (Second Edition) W3C Recommendation 6 October 2000*, http://www.w3.org/TR/2000/REC-xml-20001006

[XMLns] *Namespaces in XML* - World Wide Web Consortium 14-January-1999 http://www.w3.org/TR/REC-xml-names/

[XMLschema0] XML Schema Part 0: Primer - W3C Recommendation - 2 May 2001 http://www.w3.org/TR/xmlschema-0/

[XMLschema1] XML Schema Part 1: Structures - W3C Recommendation - 2 May 2001 http://www.w3.org/TR/xmlschema-1/

[XMLschema2] XML Schema Part 2: Datatypes - W3C Recommendation - 2 May 2001 http://www.w3.org/TR/xmlschema-2/

[XMLstylesheet] Associating Style Sheets with XML documents Version 1.0 - W3C Recommendation - 29 June 1999, http://www.w3.org/TR/xml-stylesheet/

[XSL] *Extensible Stylesheet Language* (XSL), Version 1.0 W3C Recommendation 15 October 2001 http://www.w3.org/TR/2001/REC-xsl-20011015/xslspecRX.pdf

[XSL] *Extensible Stylesheet Language*, http://www.w3.org/TR/WD-xsl

[XSLT] J.Clark: *XSL Transformation (XSLT) Version 1.0*, (1999) http://www.w3.org/TR/xslt

[XSLT] *XSL Transformations (XSLT*) Version 1.0, W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt