“XML: More than an e-publishing language”

Jaime Delgado
Universitat Pompeu Fabra
(UPF)
Barcelona (Spain)
jaime.delgado@tecn.upf.es

Ramon Martí
media technology Group
(MTG)
Barcelona (Spain)
ramon.marti@mtg.es

Xavier Perramon
Universitat Politècnica de Catalunya
(UPC)
Barcelona (Spain)
perramon@ac.upc.es

Abstract

XML is an SGML-based language designed for the interchange of documents with more flexible and powerful features than those provided by HTML. It can be considered as an intermediate step between HTML and SGML. On one hand, it is a fully conformant SGML application, but without most of the features of SGML that make it complex to handle or that are rarely used. On the other hand, XML is not restricted to a fixed DTD as HTML is: users can choose from the set of available DTDs the one that is best suited for their applications, or develop new ones for fully satisfying their requirements.

Although XML is initially a meta-language intended for publishing on the Web, there are many other applications of XML. Some of them are being widely used, some others are under investigation and some others still need be to be conceived.

In the paper, we describe some general uses of XML, such as metadata description or XML/EDI, and some experiences with XML, that include statistics handling, forms interchange, and XML/EDI payment.

1. Introduction to XML

The Extensible Markup Language (XML) [XML] is a document description language based on the syntax defined in the SGML standard, plus certain extensions known as the "Web SGML Adaptations Annex". The first version of the XML specification was published in February 1998.

Like all SGML applications, XML makes use of tags embedded in the content for describing the structure of the document. This structure has to follow the rules prescribed in a Document Type Definition (DTD).

One of the design goals of XML is to provide more flexibility than HTML, the document encoding extensively used in the World Wide Web, but without the need to support the full functionality specified in the SGML standard, which is in general considered too complex for the average requirements of typical document-based applications. Thus, XML can be viewed as an intermediate step between HTML and SGML. On one hand, it is a fully conformant SGML application, providing a subset that does not include most of the features of SGML that make it complex to handle or that are rarely used. On the other hand, XML is not restricted to a fixed document structure, as HTML is: users can choose from the set of available DTDs the one that is best suited for their applications, or develop new ones for meeting their specific requirements.

XML documents, as in HTML, are structured as a tree of hierarchical elements, delimited by means of markup tags. These tags may include attributes related to each element. However, the main difference between HTML and XML is that the former specifies a predefined document structure, determining which elements are allowed where and with which attributes, whereas the latter allows users or applications to declare their own document structures, suited to their needs. Therefore, HTML is an SGML application that uses a fixed, predetermined DTD, while in XML it may be user-defined. Put in other words, HTML defines the syntax and the semantics, and XML defines only the syntax, permitting applications to provide the semantics.

Actually, it is even possible to have an XML document without a formal DTD, as the XML specification distinguishes two levels of conformance: "well-formed" documents and "valid" documents. An XML document is well-formed if it is syntactically correct, regardless of any structural rules, so that there is no need for a DTD when checking for well-formedness. Valid documents are well-formed documents, which, in addition, follow the rules specified in the corresponding DTD. In general, most applications will provide a document structure and require the use of valid documents. If an XML document is valid, it is a conformant SGML document and can therefore be processed with a generic SGML parser. However, well-formed XML documents that are not valid violate the SGML standard.

Another remarkable difference between HTML and XML is that the former makes use of certain features from the SGML language, like tag omission or attribute minimisation, that are intended for facilitating the task of authors. This makes it easier to create HTML documents by hand, which used to be common practice in the beginnings of the World Wide Web, although this adds complexity to parsers, and it can even lead to interoperability problems. On the contrary, one of the design goals of XML is that document processing should be as simple as possible, at the expense of making the composition of documents completely by hand more cumbersome. This is not considered to be a severe problem, as it is expected that XML documents will generally be produced automatically or with the assistance of appropriate editing tools.

Since XML is simply a syntax for describing documents, and each application is free to define the semantics associated with every type of element, it may be necessary to provide additional information so that a document can be visualised with an XML-based browser. This is not needed in HTML, as browsers are supposed to know what each different type of element represents, e.g. "TITLE", "H1", "H2", "P", "A", "IMG", "FORM", etc., and how to interpret their attributes. The information describing how an XML document is to be visualised, or otherwise processed in general, can be specified with the Extensible Stylesheet Language (XSL) [XSL].

By using XSL, an XML document can be transformed into another XML document containing elements and attributes that represent formatting properties. The resulting document can then be interpreted to produce formatted output, e.g. on paper, on a display, or any other media.

The specific rules for transforming the source document into the result document are specified with the XSLT (XSL Transformations) language [XSLT]. XSLT rules associate patterns with templates, and for every element in the source document that matches a given pattern, the corresponding template is used for creating contents in the result document. Although the XSLT language was designed for adding XSL stylesheet information to documents, it can be used for making generic transformations to XML documents, independent of XSL.

When XSLT is used with XSL, the result document contains presentation information expressed in a formatting vocabulary. This vocabulary is a set of document elements and attributes that represent formatting objects and formatting properties with semantics defined in the XSL specification. The presentation information is used by browsers for determining the appearance of each element and its

contents in the presentation medium, e.g. dimensions and position of blocks, font size, type style, colour, etc.

The XML syntax is complemented by another normative specification that defines the so-called "namespaces" in XML. Namespaces allow for the use of universally unique element types and attribute names in an XML document. Uniqueness is achieved by adding a prefix to the names belonging to a namespace. Every prefix is declared locally in the document and associated with a namespace by means of a URI. Namespaces are used for avoiding name collisions when a document is to be processed by different applications. The formatting vocabulary defined by XSL is an example of an XML namespace.

Other members of the XML family of specifications include: the XML Linking Language (XLink), a language for specifying complex relationships or links between document elements; the XML Path Language (XPath) for addressing internal parts of a document; the XML Pointer Language (XPointer), based on XPath, for specifying document fragment identifiers as part of URLs; and the XML Fragment Interchange, a method for processing parts of an XML document independently of the rest of the document. The development of these specifications is carried out by the World Wide Web Consortium (W3C). Some of the latter are currently working drafts, but most of them have been formally published as official W3C Recommendations.

The W3C has also developed XHTML, a reformulation of the HTML language using the XML syntax, and the Synchronized Multimedia Integration Language (SMIL) [XML SMIL] for the description of multimedia presentations using XML.

Since the publication of the XML specifications, and even before, an important number of applications based on this language have been developed. Some are adaptations of already existing structured information representations to the new syntax, while others are completely new applications that take full benefit of the XML advantages. Fields of application are very varied, and range from specialised notations like mathematical or chemical formulae, to WML [WML], the language used by wireless hypertext browsers. One of the factors that have contributed to this rapidly growing interest on XML is undoubtedly its combination of flexibility and simplicity.

In the rest of this article we will describe some experiences with the use of the XML technologies in different applications, as an example of what can be achieved with XML.

2. XML and e-publishing

Since XML keeps almost all of the features of SMGL, XML can be used in the same way as SGML, making XML one very good format for publishing environments. Then, in the publishing area, XML should be used to provide document-oriented structure to document-oriented content. This means that XML is to be used to provide to the document content (e.g. text, images) a logical structure (e.g. chapters, paragraphs), but also formatting information (e.g. line spacing, text rendition attributes) and layout information (e.g. pagination).

Then, in publishing area, XML DTDs will be used to define the rules for the logical structure, for the generic layout structure and for formatting, through the definition of the appropriate tags and the position in the structure where they can be used. Moreover, styles can also be used in XML to provide formatting.

For presenting one XML document containing specific tags (following or not a specific DTD), it is required a viewer able to understand the meaning of these tags and able to know how to represent the associated content (e.g. for document-based XML, it must know that <bold> tag means that the associated information must be presented in bold).

Nowadays, the most common Web browsers support XML. But they do not know the meaning of the tags, so the way they show the file to the user is presenting the XML information in a more user friendly way, that is, presenting with the tags and the content, but in different colours and in an indented structure.

For all these reasons, it can be affirmed that XML is a good format to be used by e-publishing companies, which must define and use a DTD that contains all the features required for the specific publishing requirements in the company. Anyway, XML files following proprietary DTDs cannot be used for open widespread e-publishing distribution, since proprietary viewers should be required for each DTD. It is at this point when conversion to wide-used formats is required. Depending on the way the document is to be distributed, different formats can be used. Formats like PDF are oriented to non editable documents with specific formatting and layout information; formats like Microsoft Word are intended for editable documents, with logical, formatting and styles information; and formats like HTML are mainly oriented to documents not to be edited, with logical and very basic formatting information (if no styles are present, the browser can choose the way how to present to the user the different logical elements; if styles are used, complex formatting information can be defined and applied).

Among the different ways of making document format conversions, one is through XSL and XSLT. With XSL stylesheets with XSLT instructions, an XML source document can be transformed into another ASCII tag-based target format. Since usual web browsers support XSL and XSLT, one common practice is using specific XSL+XSLT stylesheets to transform XML files into HTML. The XSL file is a template specific for one DTD that include tags of the target format (HTML in our case) and also XSLT instructions for filling the template with the information extracted from specific tags of the source file (XML file in our case). Then, the browser reads the XML file, applies the XSL+XSLT transformations and then what the browser has is an HTML file, which it can present in the usual way.

3. Metadata description with XML

With the increasingly high volume of information available on the net, the problem of searching, locating and filtering this information becomes more and more non-trivial. One proposed solution to this problem is the use of metadata, that is, data about data.

Metadata consist of data with a pre-defined structure that are associated with a piece of content or, in general, a "resource" (a book, a document in the WWW, an image, a movie, etc.), and that provide information about that resource. Metadata are normally expressed as attributes or properties represented by means of name-value pairs. Examples of typical attribute names (also called "labels" or "descriptors") are: title, author, keywords, language, date of last update, references, etc. These attributes can be used for resource searching, but also for other applications like document administration and archival, rights management, content rating, security and access control, etc.

There are two aspects to consider about metadata specifications. One is the set of attributes that are defined for a particular application, and the other is the representation or encoding format for these attributes. The former is of special relevance to interoperability, an the latter is an important issue when metadata are to be machine-interpreted.

Several metadata sets have been developed for use in different environments, like the Dublin Core for document management and cataloguing, PICS (Platform for Internet Content Selection) for content rating in the WWW, or MPEG-7 for multimedia content description. With regard to the representation, there are also different techniques. When applied to resources accessible through the WWW, it is usual to encode metadata attributes in the HTTP protocol headers. If the resource is an HTML document, these attributes can also be conveyed in the document header, as "META" elements.

However, XML offers a more flexible and powerful method for specifying metadata properties. For example, the MPEG-7 Description Definition Language (DDL), currently under development, makes use of the XML syntax. And the W3C has published an official Recommendation defining the Resource Description Framework (RDF), a general model for expressing metadata with XML.

In the RDF terminology, metadata attributes associated with a resource are called "properties", each consisting of a name and a value. A property, together with the resource it refers to, forms a "statement", where the resource is termed the "subject", the property name is termed the "predicate", and the property value is termed the "object" of the statement. An object, i.e. a property value, can be expressed either as a literal or as another resource, which may have other properties associated with it. For example, if we consider a given document as a resource, it could have properties such as "author", "owner", "addressee", etc. The value of each of these properties could be directly a literal, e.g. a name identifying a person, or another resource having properties such as "name", "company", "address", etc.

A set of RDF properties associated with a resource is represented as an XML element called "Description". The resource is identified by a URI, and the properties are sub-elements of the description, which may have content directly representing the value, or an XML attribute referencing another resource. All element names used in the description must belong to an XML namespace, which has to be properly declared.

As an example of a practical application of RDF, we have defined a set of security attributes for use in the ACTS project TRADE (Trials in the Domain of Electronic Commerce) [TRADE]. These attributes are associated with resources stored in a server, such as documents or electronic commerce transactions, and represent information about the owner of the resource, access rights, authenticity, integrity, privacy, etc.

4. XML and EDI

Electronic commerce is one of the main applications that have taken advantage of the data structuring and processing capabilities of XML. Several initiatives have been undertaken in this field, the most remarkable being XML/EDI, which combines the widely deployed EDI technology with the versatility of the XML language. This combination provides a standard framework to describe different types of business data, so that the information, whether in a transaction, catalog, etc., can be searched, decoded, manipulated, and displayed consistently and correctly.

The benefits of using XML for the interchange of transactions extend beyond the simple structuring of the messages (which is something that EDI itself already provides). The technologies developed around XML make it possible, for example, to define rules for the processing of messages, including display and presentation, by means of XSL. XML also allows for references to external resources, which may contain repositories of DTDs, syntactical rules, scripts, etc., thus extending the functionality of EDI dictionaries. In addition, applications can be integrated with Web-based

systems using HTTP, metadata, search engines, data manipulation robots, and a host of new technologies which are continuously evolving.

The XML/EDI Group, set up in July 1997 with the participation of a large number of companies, published in 1998 the "Guidelines for using XML for Electronic Data Interchange" [XMLEDI Guidelines], currently the main XML/EDI reference, which will eventually form the basis of a formal "Specification of an EDI Application for XML". Since then, other fora have also been developing guidelines for the use of XML/EDI. One of the most active groups in this area is the XML/EDI Workshop of the CEN/ISSS (Information Society Standardization System of the European Committee for Standardization). In early 2000, this group published a CEN Workshop Agreement (CWA), based on the results of the ISIS European XML/EDI Pilot Project.

The XML/EDI CWA contains a series of recommendations on technology issues, like the use of XSL for creating XML/EDI messages, the use of XSLT for producing different outputs from a single XML input, the inclusion of external non-XML data in an XML/EDI message, or the possibility to digitally sign parts of XML documents. The CWA also contains recommendations on best practices for XML/EDI applications, regarding interoperability, testing and evaluation. Finally, there is a third group of recommendations on XML/EDI integration, unification and harmonisation issues.

Apart from XML/EDI, other electronic commerce applications based on XML have been developed or are currently under development. These include Lite-EDI, SIMPL-EDI and ebXML [XML EBXML]. Lite-EDI is designed for the interchange of EDI messages over non-dedicated networks such as the Internet, making use of simple message structures, with as few optional elements as possible. SIMPL-EDI is a simplified subset of the UN/EDIFACT standard for EDI messages that includes support for the most commonly used types of messages like invoice, purchase order and response, dispatch advice, and payment. XML DTDs have been defined for these messages in the framework of the CEN/ISSS XML/EDI Pilot Project. Finally, ebXML is a joint initiative by UN/CEFACT (United Nations Centre for Trade Facilitation and Electronic Business) and OASIS (Organization for the Advancement of Structured Information Standards) [XML OASIS] for standardising e-business applications based on XML.

5. XML applications

5.1. XML/EDI for payment

In the context of the above-mentioned TRADE project, we have developed an application of XML for business-to-business interchange using XML/EDI. The result is the implementation of an XML/EDI payment system. In this system, the provider requests the creation of an invoice, edits it and sends it to the customer, who, after performing the payment, informs the provider, who finally generates a receipt as a proof of the operation. By design of the project, all interactions between provider and customer take place through a central server.

The basic types of messages that we have used in our implementation are the invoice and the receipt. For each of these, we define a partial message type, which contains template information and is presented to the user for filling in, and a complete message type, which includes the information entered by the user.

Different processes are carried out in our payment mechanism. On the server side, XML_EDI_PARTIAL messages (either INVOICE or RECEIPT) are created and sent to the clients, and XML_EDI_COMPLETE messages are validated. On the clients side (provider and customer),

XML_EDI_PARTIAL messages received from the server are edited, and XML_EDI_COMPLETE messages are submitted back to the server and visualised.

The implementation started with a search for the required EDI types of messages. The SIMPL-EDI INVOICE message was used for representing invoices. However, no message type was found suitable for receipts, so that we defined our own XML document structure for representing them.

Once the XML messages to be interchanged were chosen, the next step was the design of XSL templates for transforming these XML messages into HTML documents, which can be viewed with a Web browser. The following templates were created: XSL for Viewing Invoice, XSL for Editing Invoice, XSL for Viewing Receipt, and XSL for Editing Receipt.

The processes that run on the server were implemented using servlets. A servlet is a program written in Java (equivalent to the traditional CGI programs) that runs on the server and returns a document (HTML, XML) to the browser.

The following goals have been achieved with our XML-based payment implementation: facilitating the incorporation of new messages, and the portability and feasibility of replacing a traditional electronic commerce system (based on the execution of CGIs on the server) by our modules (based on the interchange of XML documents).

5.2. Forms interchange

Again in the TRADE project, we have developed and electronic commerce application for the provision of multimedia services [TRADE Legal] [TRADE LegalWeb]. In particular, we have focussed on legal and administrative services. In this last group of services, it is very often necessary to produce and interchange forms, which, most of the times, should follow an official format. In the first phase of the project, forms sending had been implemented using HTML based forms and an .asp page that generated a file containing the values of the fields in the form.

This approach has some problems, such as:

- HTML permits very basic forms construction.
- HTML cannot implement relationships between form values.
- The file generated for storing form values is a basic text file with format "Name, Value"

A solution to these problems is the use of XML to construct and store the forms in an “intelligent” manner, since XML is able to give semantics to forms.

This application is designed to create, edit and transfer forms using XML documents starting from forms on paper. These forms conserve the original structure maintaining all their semantic characteristic. A form editor creates the forms using a graphic interface.

A form is composed by four documents:

- A DTD document that defines the form structure.
- An XML document that contains the data introduced by the user.
- An XSL document that defines the data visualisation style.
- An XML style document to show the data contained in the general document.

The forms are composed by four different field types:

- Single. It is a blank space to be filled by the user.
- Compound. It is composed by several fields of any type. It can be seen as few fields grouped.

- Enumerated. This field only can take a specific value.
- Multi-enumerated. This field can take various discrete values.

A field can be mandatory or not.

This application has been developed in Java language. The data access is ODBC (Open DataBase Connection) in order to obtain the most independence between the data base and the application.

Every form is internally represented by a tree and every field by a node. The user can select the required node when he uses the application to edit or create a new field in a form. The user can see on the left the tree structure of the form and the selected field properties on the right. The use is very intuitive.

5.3. Statistics Handling

In the Statistics Handling application, XML was used to store structured information about statistics on Internet files access. For this reason, an XML DTD was defined with all the fields and the logical structure required for storing Internet access statistics information.

In principle, tags were defined to be easily user-readable, but XML statistics files become big and not as user-friendlier as wished. To present the information in a user-friendly way, it is necessary to have an application that knows the tags in the file and its meaning (that is, knowing the DTD, if it exists) and that also knows the way to represent in a user-friendly way all files following that DTD.

The solution adopted to present the information to the user was to convert the XML file into HTML, so usual web browsers could be used as viewers. It must be noticed that the XML statistics file is information-oriented, and HTML is document-oriented. In this sense, XML and HTML are complementary: XML was used to structure the information, and HTML to present it in a user-friendly way.

XSLT is a part of the XSL standard that defines the way to do transformation between XML source format and another ASCII tag-based target format. This is done through the definition of XML template files, that include tags of the target format (HTML in our case) and also instructions for filling it with the information extracted from specific tags of the source file (XML file in our case).

5.3.1. Statistics Report Files

This example describes the different file types that were used in the statistics handling application based on XML:

- DTD
- XML
- XSL + XSLT
- HTML

5.3.1.1. DTD

One DTD of the Statistics handling application was defined (see Annex 1, Example DTD). In this DTD it was defined all the required fields for a statistics information report and the logical structure of these elements.

5.3.1.2. XML

In "Annex 1, XML Example", there is one XML document containing one specific statistics report for the Example user. It can be seen how this XML report follows the logical structure defined in "Example.dtd" through the use of the tags defined in this DTD.
It can also be seen that this report makes reference to "Example.xsl" stylesheet, that means that this report is to be represented using the rules in the stylesheet.

5.3.1.3. XSL + XSLT Stylesheet

In order to present statistics information in a user-friendly way, XSL stylesheets with XSLT commands were defined and used to transform a web statistics information from a document in XML format into HTML.

In "Annex 1, Example XSL+XSLT" there is the XSL file used, that includes XSL tags together with XSLT transformation commands (in bold) which are used to get information from specific tags of the XML document they are being applied. HTML tags (in plain text) are inside the file, in order to provide the HTML logical structure and formatting to the extracted text.

5.3.1.4. HTML Report (XML + XSL)

In our application, when using an XSL with XSLT commands to represent the XML file, transformations were done to create an HTML document. In "Annex 1, HTML Example" it can be seen the result of applying the Example.xsl stylesheet to the Example.xml file, that is an HTML file. In this file it can be seen how the HTML tags that were in the Example.xsl file are present in this file, and that the XSLT commands have been substituted by the corresponding text extracted from the Example.xml file.

6. Conclusions and Future work

We have introduced the XML standard and technology, including various different applications we have developed. Although XML is based in an old technology, SGML, it can be seen as a new way to do electronic publishing, and other applications, on the Web, and on other environments.

HTML (1996) and SGML (1986) are not practical for web structured documents, since HTML has fixed semantics and not arbitrary structure, and SGML provides arbitrary structure and it is difficult to implement for web applications.

XML (1998) is an SGML application profile: SGML subset for the interchange of structured documents over Internet. It provides the rules for defining and using ASCII tag-based logically structured document templates for Internet interchange of ASCII content.

XML inherits the best features of:

- SGML: Information structure for publications
- HTML: Document structure for the Web.
- XML: Information structure for the Web.

Related standards, such as XSL or XSLT provide extended features to the initial XML.

XML has been successfully used in several experiences, from general ones such as metadata description or XML/EDI, till specific ones such as forms interchange or statistics handling.

There are however other innovative uses of XML, most of them with specified DTDs, such as:

- Accounting:
  - XML Financial Reporting Markup Language (XFRML) (http://www.xfrml.org/)
- Banking:
  - Interactive Financial Exchange (IFX) (http://www.ifxforum.org/)
- Communications:
  - Telecommunications Interchange Markup (TIM) (http://www.atis.org/atis/tcif/ipi/dl_tim.htm);
  - Wireless Markup Language (WML) (http://www.wapforum.org/what/technical.htm)
- Directory Services:
  - Directory Services Markup Language (DSML) (http://www.dsml.org/);
  - DirXML (http://www.novell.com/products/nds/dirxml/)
- Electronic Commerce:
  - Product Data Markup Language (PDML) (http://pdit.com/pdml/);
  - Business Rules Markup Language (BRML) (http://www.research.ibm.com/rules/home.html);
  - Commerce XML (cXML) (http://www.cxml.org/home/);
  - Open Trading Protocol (OTP) (http://www.otp.org/)
- News:
  - XML News Industry Text Format (XML NITF) (http://www.iptc.org/iptc/xmlnitf.htm);
  - XMLNews (XML-Story, XMLNews-Meta) (http://www.xmlnews.org/)
- Science:
  - Astronomical Instrument Markup Language (AIML) (http://pioneer.gsfc.nasa.gov/public/aiml/);
  - Molecular Dynamics Markup Language (MoDL) (http://violet.csa.iisc.ernet.in/~modl/);
  - OpenMath (http://www.openmath.org/omsoc/);
  - BIOpolymer Markup Language (BIOML) (http://204.112.55.140/BIOML/);
  - Chemical Markup Language (CML) (http://www.xml-cml.org/);
  - Bioinformatic Sequence Markup Language (BSML) (http://www.visualgenomics.com/bsml/);
  - Mathematical Markup Language (MathML) (http://www.w3.org/TR/REC-MathML/)
- Software:
  - Bean Markup Language (BML) (http://www.alphaworks.ibm.com/formula/bml);
  - Koala Bean Markup Language (KBML) (http://www.inria.fr/koala/kbml/);
  - Open Software Description Format (OSD) (http://www.w3.org/TR/NOTE-OSD.html);
  - XML Metadata Interchange (XMI) (http://www.omg.org/news/pr99.html#xmi)
- Travel:
  - Open Travel Alliance (OTA) (http://www2.air-transport.org/ota/)
- User Interface:
  - Extensible User Interface Language (XUL) (http://www.mozilla.org/xpfe/xptoolkit/xulintro.html);
  - User Interface Markup Language (UIML) (http://uiml.org/specs/index.html)
- Web Applications:
  - Cold Fusion Markup Language (CFML) (http://www1.allaire.com/documents/cf4/);
  - Extensible Log Format (XLF) (http://www.docuverse.com/xlf/);

    - World Wide Web Distributed Authoring and Versioning (WebDAV) (http://www.ics.uci.edu/pub/ietf/webdav/)
- Workflow:
    - Simple Workflow Access Protocol (SWAP) (http://www.ics.uci.edu/~ietfswap/);
    - Workflow XML: (Wf-XML) (http://www.aiim.org/wfmc/)
- IPR (Intellectual Property Rights) transfer and negotiation
- etc.

7. References

[ACTS] ACTS (Advanced Communications Technology and Services) (http://www.infowin.org/acts/.)

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

[SGML] ISO. International Standard ISO 8879:1986, Information Processing -- Text and office systems -- Standard Generalized Markup Language (SGML), 1st edition, 1986

[TRADE Legal] LLORENTE S. and DELGADO J. "Legal and Administrative Services through Electronic Commerce", 7th International Intelligence in Services and Networks Conference (IS&N'2000), 23-25 February 2000, Athens (Greece).

[TRADE LegalWeb] TRADE Legal and Administrative applications Web Page, http://www.mtg.es/trade/.

[TRADE WEB] TRADE Project Web Page (http://trade.cosi.it/)

[XML] T. Bray, J. Paoli, C. M. Sperberg-McQueen, editors. Extensible Markup Language (XML) 1.0, W3C Recommendation, 1998 (http://www.w3.org/TR/REC-xml)

[XML Namespaces] T. Bray, D. Hollander, A. Layman, editors. Namespaces in XML, W3C Recommendation, 1999 (http://www.w3.org/TR/REC-xml-names)

[XML EBXML] ebXML (http://www.ebxml.org/)

[XML FAQ] The XML FAQ (http://www.ucc.ie/xml)

[XML W3C] XML - W3 consortium (http://www.w3.org/XML)

[XMLINFO] XMLINFO (http://www.xmlinfo.com)

[XML OASIS] Organization for the Advancement of Structured Information Standards (http://www.oasis-open.org)

[XML SMIL] Synchronized Multimedia Integration Language (http://www.w3.org/TR/WD-smil/)

[XML.com] XML.com (http://www.xml.com)

[XML.org] The XML industry portal (http://www.xml.org)

[XML Murray] P. Murray-Rust, "XML: The thinking site's HTML", in: The WOT Journal (Web Open Technologies), vol. 1, issue 5, October 1997

[XMLEDI EEMA] EEMA EDI/EC Work Group (http://www.edi-tie.nl/edifact/xml-edi.htm)

[XMLEDI European Pilot] European XML/EDI Pilot Project (http://www.cenorm.be/isss/workshop/ec/xmledi/isss-xml.html)

[XMLEDI Group] XML/EDI Group (http://www.geocities.com/WallStreet/Floor/5815/)

[XMLEDI Guidelines] Guidelines for using XML for Electronic Data Interchange (http://www.geocities.com/WallStreet/Floor/5815/guide.htm) (http://www.xml.com/pub/r/Guidelines_for_using_XML_for_Electronic_Data_Interchange)

[XMLEDI X12] ANSI ASC X12/XML (http://www.disa.org/x12/)

[XMLEDI Webber] D. Webber, "XML/EDI frameworks", in: The WOT Journal (Web Open Technologies), vol. 1, issue 5, October 1997

[XSL] XSL - W3 consortium (http://www.w3.org/Style/XSL)

[XSL XMLINFO] XSL - XMLINFO (http://www.xmlinfo.com/xsl/)

[XSLT] J. Clark, editor. XSL Transformations (XSLT) version 1.0, W3C Recommendation, 1999 (http://www.w3.org/TR/xslt)

Annex 1: Example

This annex presents one example, with one sample of each one of different XML-related files. In this case, they are related to the statistics handling application based on XML that is described in the paper. The following are the files in this annex:

- DTD: Example.dtd
- XML: Example.xml
- XSL: Example.xsl
- HTML: Example.html

A1.1. Example DTD (Example.dtd)

```
<?XML version="1.0" encoding="iso-8859-1"?>
<!ELEMENT ExampleReport (ExampleHeader, TimeSpan, TopObjectList)>
<!ELEMENT ExampleHeader (MoDAnbNumber, Version, Format)>
<!ELEMENT MoDAnbNumber (#PCDATA)>
<!ELEMENT Version (#PCDATA)>
<!ELEMENT Format (#PCDATA)>
<!ELEMENT TimeSpan (FromDate, ToDate, TodayDate)>
<!ELEMENT FromDate (#PCDATA)>
<!ELEMENT ToDate (#PCDATA)>
<!ELEMENT TodayDate (#PCDATA)>
<!ELEMENT TopObjectList (TopObject)+>
<!ELEMENT TopObject (Position, InitIdentifier, MMType, Quantity)>
```

```
<!ELEMENT Position(#PCDATA)>
<!ELEMENT InitIdentifier(#PCDATA)>
<!ELEMENT MMType(#PCDATA)>
<!ELEMENT Quantity(#PCDATA)>
```

A1.2. Example XML (Example.xml)

```
<?XML version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE ExampleReport SYSTEM "Example.dtd">
<?XML:stylesheet href="Example.xsl" type="text/xsl"?>
<ExampleReport>
      <ExampleHeader>
            <MoDAnbNumber>0000900001</MoDAnbNumber>
            <Version>01</Version>
            <Format>MODKONTR</Format>
      </ExampleHeader>
      <TimeSpan>
            <FromDate>19980101</FromDate>
            <ToDate>19981231</ToDate>
            <TodayDate>20000601</TodayDate>
      </TimeSpan>
      <TopObjectList>
            <TopObject>
                  <Position>1</Position>
                  <InitIdentifier>123</InitIdentifier>
                  <MMType>1</MMType>
                  <Quantity>123</Quantity>
            </TopObject>
            <TopObject>
                  <Position>2</Position>
                  <InitIdentifier>234</InitIdentifier>
                  <MMType>1</MMType>
                  <Quantity>99</Quantity>
            </TopObject>
      </TopObjectList>
</ExampleReport>
```

A1.3. Example XSL + XSLT Stylesheet (Example.xsl)

```
<?XML version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet                    XMLns:xsl="http://www.w3.org/TR/WD-xsl"
XMLns="http://www.w3.org/TR/REC-html40" result-ns="">
<xsl:template match="/">
<HTML> <BODY><CODE>
[VERSION:                          V.                          <xsl:value-of
select="ExampleReport/ExampleHeader/Version"/>]<BR/>
Timespan:   <xsl:value-of   select="ExampleReport/TimeSpan/FromDate"/>   to
<xsl:value-of select="ExampleReport/TimeSpan/ToDate"/><P/>
</CODE></BODY></HTML>
<U>Top 100 most requested objects</U><BR/>
<TABLE BORDER="0">
<TR><TD width="50"><U>Position</U></TD>
      <TD width="50"><U>Init Id</U></TD>
      <TD width="150"><U>Mm-type</U></TD>
      <TD width="50"><U>Quantity</U></TD>
</TR>
<xsl:for-each select="ExampleReport/TopObjectList/TopObject">
<TR><TD><xsl:value-of select="Position"/></TD>
      <TD><xsl:value-of select="InitIdentifier"/></TD>
      <TD><xsl:value-of select="MMType"/></TD>
```

```
      <TD><xsl:value-of select="Quantity"/></TD>
</TR>
</xsl:for-each>
</TABLE><BR/>
</xsl:template>
</xsl:stylesheet>
```

A1.4. Example HTML (XML + XSL) (Example.html)

```
<HTML> <BODY><CODE>
[VERSION: V. 1.00]<BR/>
Timespan: 2.11.199 to 8.11.1998<BR/>
<U>Top 100 most requested objects</U><BR/>
<TABLE BORDER="0">
<TR><TD width="50"><U>Position</U></TD>
      <TD width="50"><U>Init Id</U></TD>
      <TD width="150"><U>Mm-type</U></TD>
      <TD width="50"><U>Quantity</U></TD>
</TR>
<TR><TD>1</TD>
      <TD>123</TD>
      <TD>1</TD>
      <TD>123</TD>
</TR>
<TR><TD>2</TD>
      <TD>234</TD>
      <TD>1</TD>
      <TD>99</TD>
</TR>
</TABLE><BR/>
</CODE></BODY></HTML>
```