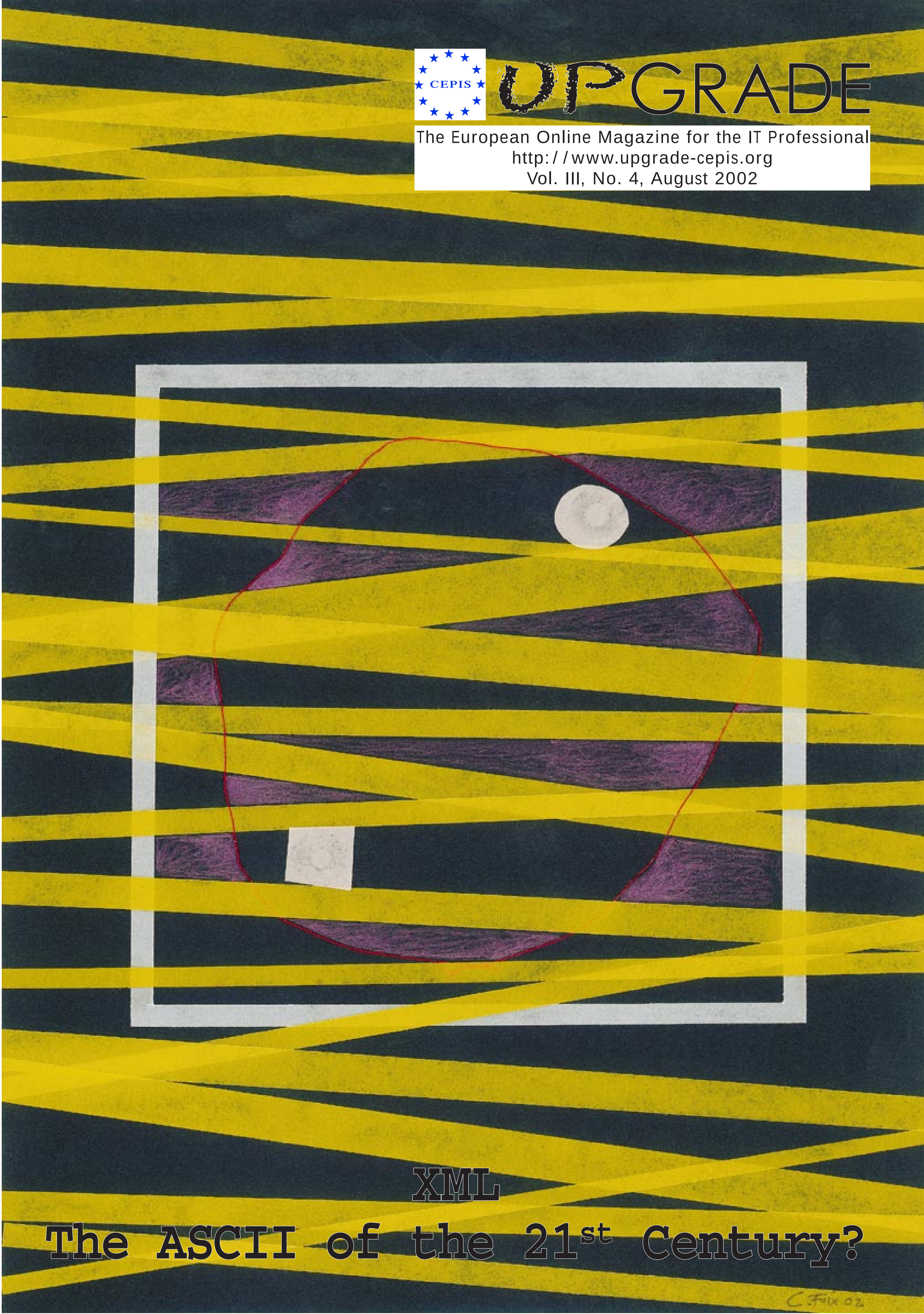
CEPIS

UPGRADE

The European Online Magazine for the IT Professional
http://www.upgrade-cepis.org
Vol. III, No. 4, August 2002



XML
The ASCII of the 21st Century?

EUropean Certification of Informatics Professionals

An accepted European
ICT certification standard

<http://www.eucip.com/>

UPGRADE
The European Online Magazine for the IT Professional
http://www.upgrade-cepis.org
Vol. III, No. 4, August 2002

UPGRADE is the European Online Magazine for the Information Technology Professionals, published bimonthly at <http://www.upgrade-cepis.org/>.

Publisher
UPGRADE is published on behalf of CEPIS (Council of European Professional Informatics Societies, <http://www.cepis.org/>) by NOVÁTICA <http://www.ati.es/novatica/>, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática <http://www.ati.es/>).
UPGRADE is also published in Spanish (full issue printed, some articles online) by NOVÁTICA, and in Italian (abstracts and some articles online) by the Italian CEPIS society ALSI <http://www.alsi.it> and the Italian IT portal Tecnoteca <http://www.tecnoteca.it/>.
UPGRADE was created in October 2000 by CEPIS and was first published by NOVÁTICA and INFORMATIK/INFORMATIQUE, bimonthly journal of SVI/FSI (Swiss Federation of Professional Informatics Societies, <http://www.svifsi.ch/>).

ISSN 1684-5285

XML (eXtensible Markup Language): The ASCII of the 21st Century?

Guest Editors: Luis Sánchez-Fernández and Carlos Delgado-Kloos

Joint issue with NOVÁTICA

Coming issue:
"**Artificial Intelligence**"

Editorial

EUCIP, a Certification Service for the European Informatics Professionals promoted by CEPIS

Dear readers of UPGRADE,

As many of you will surely know, CEPIS (Council of European Professional Informatics Societies <http://www.cepis.org>), created in 1991, unites 32 Informatics Professional Societies across Europe and is therefore the voice of more than 150,000 ICT Professionals in our continent. Some of you will also be aware that one of main strategic objectives of our organization reads as follows:

"To promote acceptance of professional standards for ICT Professionals. Through these, CEPIS enhances ICT professionalism and the role that ICT plays in modern society."

In order to achieve it CEPIS has developed a pan-European certification scheme: EUCIP (European Certification of Informatics Professionals). Taking advantage of existing schemes from CEPIS member societies, experiences (e.g. ISM by the British Computer Society, BCS), the objective of EUCIP is to offer certification of ICT competence at the intermediate education level (typically 1-2 years of post-tertiary education) to ensure a common quality standard which is accepted by industry, governmental and public organisations.

EUCIP will increase the market value of certified ICT practitioners, and will contribute to closing the European competency gap by attracting new practitioners to the ICT field. The standard is defined by the EUCIP syllabus.

EUCIP offers testing, not learning; i.e., it is not a learning scheme, but offers certification of ICT competence in collaboration with learning providers such as universities, private educational institutions, IT-vendors and e-learning schemes. However, certification does not require that such learning schemes are followed, and may be done by testing at any one of numerous test centres around Europe.

The initial steps of EUCIP seem quite promising:

- Ten CEPIS member societies from eight European countries are already involved in the project
- A syllabus has been developed, describing the three main knowledge areas (plan, build and operate).The syllabus is available on the public domain.
- The Question and Test Base is complete, including approximately 270 questions.
- A pilot test system has been developed, and is being evaluated by participating learning providers.
- The web site is complete, including a download area.
- Office accommodation is available in Swindon, where the EUCIP office is planned to be based.
- 17 learning providers from Ireland, Germany, Greece, Norway, Italy and the UK have joined the pilot programme.

EUCIP encompasses goals, tools and target audience which complement another successful CEPIS project, the ECDL (European Computer Driving Licence), an internationally recognised computer skills certification programme spanning some sixty countries worldwide (see <http://www.ecdl.com> for details), first implemented in Finland in 1994 and which is aimed to the end user of computers and networks.

We in CEPIS see EUCIP as another important milestone for the visibility and practical usefulness of our organization for the European ITC practitioners, in synergy with other CEPIS projects (including UPGRADE, of course!)

Let me close by inviting you all to know more about EUCIP visiting <http://www.eucip.com>, where a core level demo test is available; you might wish to try it!

Cordially,

Prof. Wolffried Stucky
President of CEPIS
Editorial Board, UPGRADE
<stucky@aifb.uni-karlsruhe.de>

Presentation

XML, Panoramic of a Revolution

Luis Sánchez-Fernández and Carlos Delgado-Kloos

We believe that *Novática* and *Upgrade* have made an excellent decision to dedicate a monograph to XML (*eXtensible Markup Language*). Since it first came on the scene, promulgated by the *World Wide Web Consortium* (W3C) in the late 90s, XML has caused a revolution in the IT world. XML is being applied in a great many fields and for various purposes: electronic archive and contents management, web publishing, interchange of electronic documents, internal format of tools, software, e-commerce, education, and many more fields too numerous to mention here. Among the less obvious fields are: chemistry, biology, theology, tourism, law, and health. For a broad overview of XML's fields of application we recommend Robin Cover's excellent work of compilation, which can be seen at <*http://xml.coverpages.org/*>.

Although, as we have said, XML is a standard developed by W3C, its applications go beyond what can be strictly speaking considered as the scope of the Web. In this sense we could very well have entitled our presentation of the monograph *"XML, not just a Web standard"*, to reflect the fact that it is a standard which started its life in the Web world but is used both within and outside the Web environment. XML is a way to represent data which is to circulate on the internet and is therefore not necessarily linked to its presentation on a browser.

In this monograph on XML we have included both technical articles and articles referring to case studies involving the use of XML in different fields.

The monograph begins with an introductory article *"XML: The ASCII of the 21st Century"* which includes a brief history, a presentation of the XML language and some of the most important related standards and a discussion about why XML has been so successful.

Following that comes an article entitled *"XML, the Development of New Business Applications and the Software Industry"*, which in part complements the previous article. This article looks at XML from a business point of view and talks about the tools which are being developed by different software companies to support the standard.

Next come several technical articles presenting XML based developments or technologies: *"Application of XML Mark-up Languages to Software Development"*, *"Querying the Semantic Web: Feasibility Issues"* and *"Digital Signature and Encryption with XML"*.

The following two articles could be described as case studies, one concerning digital television (*"Realities and Possibilities of XML in the Standardisation of Digital TV with MHP (Multimedia Home Platform)"*) and another about journalism (*"Application of XML to the Journalism Field"*).

Finally we have included three articles selected from the XML Europe 2002 Conference, the most important conference in the field of XML in Europe. The first article, *"A New Face for each Show: Make up your Content by Effective Variants Engineering"* deals with the reuse of contents by managing the different versions (variants) of the same basic content unit (information object). The second article *"Business Maps: Topic Maps go B2B"*, deals with the definition of correspondences between different ontologies in the field of B2B. The third article, *"XML and Word and XML: Downcasting and Upcasting XML Documents"* deals with conversion between RTF and

The Guest Editors

Luis Sánchez-Fernández is a Full Professor in the Department of Telematics Engineering of the Universidad Carlos III of Madrid (Spain). He is a Doctor of Telecommunications Engineering from the Universidad Politécnica de Madrid. He is the author of more than 30 publications in journals and national and international conferences. He has taken part in several national and international research projects. Currently, his main interest is in the following fields: electronic publishing, electronic journalism and XML. He is member of ATI (Asociación de Técnicos de Informática, Spain.) <luis@it.uc3m.es>

Carlos Delgado-Kloos is presently a Full Professor at the Universidad Carlos III of Madrid (Spain), where he also holds the posts of Director of the Department of Telematics Engineering and Director of the Master in E-commerce. His current interests include design languages and techniques, as well as Internet-based applications such as electronic publishing, tele-education and e-commerce. He has been involved in many projects with European, national (Spanish Ministry) and bilateral (Spanish-German and Spanish-French) funding. He has published over 60 articles in national and international conferences and journals, and has also written one book and co-edited four more. He holds or has held various posts in national and international bodies such as: vice-president of IFIP TC 10, secretary of IFIP WG 10.5, editor of the Springer journal 'Formal Aspects of Computing', assistant director of Telecommunication Engineering at his University and manager of the National Programme for Information and Communication Technologies at the Spanish Ministry for Science and Research. He has been programme committee member or chair at more than 40 conferences and workshops, including vice-programme chair for the IFIP'92 World Computer Congress and programme chair for DATE 2002. He belongs to several professional associations in Spain and abroad, ATI being one of them, and has published several papers in Novática, ATI's journal whose Editorial Board he was a member of.
<cdk@it.uc3m.es>

XML formats, based on attaching a structural significance to Microsoft Word styles.

Any reader interested in reading more on the subject of this monograph can do so by visiting some the Web pages devoted to XML which we supply links for in the following section (see “Useful references to XML”). We also give references to two books out of the many texts on XML which are constantly coming out (a search at <http://www.amazon.com> for books about XML yielded 359 hits), and provide a glossary of XML terms.

We do not want to close without expressing our thanks to all the authors of the articles in this monograph for their contributions. Most of the authors belong to the XML-ES research network, partially subsidised by the Spanish Ministry of Science and Technology through special funding. We would also like to express our thanks for the funding of this initiative. Finally, we would like thank the editors of Upgrade, Rafael Fernández Calvo and François Louis Nicolet, for having offered to publish this monograph and for all the help they have given us.

Note:

This monograph will be also published in Spanish (full issue printed, some articles online) by Novática, journal of the Spanish CEPIS society ATI (Asociación de Técnicos de Informática) at <http://www.ati.es/novatica/>, and in Italian (online edition only, containing abstracts and some articles) by the Italian CEPIS society ALSI and the Italian IT portal Tecnoteca at <http://www.tecnoteca.it>.

Translated by Steve Turpin

Useful References on XML

Note: see also the References included in the papers making part of this issue.

Main conferences

XML Europe: <http://www.xmleurope.com>
XML Conference & Exposition: <http://www.xmlconference.org/xmlusa/>

Books

Richard Anderson, Mark Birbeck, Michael Kay, Steven Livingstone, Brian Loesgen, Didier Martin, Stephen Mohr, Nikola Ozu, Bruce Peat, Jonathan Pinnock, Peter Stark, Kevin Williams: Professional XML, Wrox Press Inc; ISBN: 1861003110.
Benoit Marchal: XML by Example, Que; ISBN: 0789722429.

Websites

Extensible Markup Language (XML). World Wide Web Consortium: <http://www.w3.org/XML>
Oasis consortium: <http://www.oasis-open.org/>
XML.org : <http://www.xml.org/>.
O’Reilly XML.com: <http://www.xml.com/>
XML Journal. SYS-CON Media: <http://www.sys-con.com/xml>
Xmlu.com. Division of Architag International: <http://www.xmlu.com/>
XML Files. INT Media Group, Incorporated. <http://www.xmlfiles.com/>
Peter Flynn. The XML FAQ: <http://www.ucc.ie/xml>
Joaquín Bravo, Daniel Rodríguez, David Carrero and Alex Morales. Programming in Spanish. XML: <http://www.programacion.com/direcciones.php?categoria=xml>
XML-ES (Extensible Markup Language in Spanish). Department of Telematics Engineering and Department of Librarianship and Documentation. Carlos III University of Madrid: <http://xml.it.uc3m.es>
XMLephant, the big XML resource. Cardboard String Media: <http://www.xmlephant.com/>
DevX XML Home. DevX Inc: <http://www.xml-zone.com/>
Web Developers Virtual Library. XML. INT Media Group, Incorporated: <http://wdvl.com/Authoring/Languages/XML>
Elliotte Rusty Harold. Cafe con Leche XML News and Resources: <http://www.ibiblio.org/xml>
IEEE Internet Computing Online. XML Resources Page: <http://computer.org/internet/xml>
XML Web Ring. Web Ring Inc.: <http://b.webring.com/hub?ring=xml>
XML newsgroup: <news:comp.text.xml>
Netscape XML Developer Central: <http://developer.netscape.com/tech/xml/index.html>
Java and XML. Sun Microsystems, Inc: <http://java.sun.com/xml/>
IBM XML Zone: <http://www.ibm.com/developer/xml/>
Microsoft XML Developer Center: <http://msdn.microsoft.com/xml/default.asp>
Software AG. The XML Company: <http://www.softwareag.com/>
W3Schools: <http://www.w3schools.com/xml/>

Glossary

Application XML:
: an XML language defined by a DTD or an XML Schema used in a particular environment.

CSS:
: Cascading Style Sheets. Stylesheets which allow us to define how to present XML documents and HTML documents too.

DOM:
: Document Object Model. This is an independent interface between the platform and the programming language used which allows us to access and modify the content or structure of XML and HTML documents.

DTD:
: Document Type Definition. This is a format allowing us to define the structure and elements of a particular XML application.

FTP:
: File Transfer Protocol. This is a protocol for the transfer of files through a computer network.

HTML:
: HyperText Markup Language. This is a markup language used to create documents which are to be published on the Web. Standardised by the W3C.

HTTP:
: Hypertext Transfer Protocol. This is an application level protocol for the interchange of hypermedia information.

Markup language:
: A language which allows the addition of marks to a text document to give semantic meaning or describe how to present the content of the document.

Metadata:
: This is data used to describe or add information to other data.

Metalanguage:
: In the IT world it is a computer language used to define other languages.

Namespace:
: Allows us to identify a source which defines a set of elements and attributes used in an XML document.

RPC:
: Remote Procedure Call. This is a protocol which allows two applications running on computers connected through a network to communicate with each other by imitating the procedure call of high level programming languages.

SAX:
: Simple API for XML. This is an interface which allows us to access the content and structure of XML documents. It is one of the two analysers generally used for developing XML based software tools, the other one being DOM. SAX is not a W3C standard, but it is a "de facto" standard.

SGML:
: Standard Generalized Markup Language. Metalanguage for defining markup languages; the forerunner of XML.

SOAP:
: Simple Object Access Protocol. This is a protocol based on XML for remote method invocation via messages.

SMTP:
: Simple Mail Transfer Protocol. This is a protocol for interchanging e-mail messages.

UDDI:
: Universal Description, Discovery and Integration. This is an initiative for creating a platform-independent, open framework for describing services, discovering businesses and integrating services offered by companies over the Internet.

WML:
: Wireless Markup Language: A language which is currently used for presenting hypertext content on mobile phones and PDAs.

WSDL:
: Web Services Description Language.

W3C:
: World Wide Web Consortium. This is the organisation responsible for developing Web related standards.

XLink:
: XML Linking Language. Language for defining links between XML documents.

XML:
: Extensible Markup Language. It is a metalanguage for defining markup languages. Standardised by the W3C.

XML Schema:
: Like DTDs, it allows us to define what a particular XML application is like, only much more precisely.

XPath:
: XML Path Language. Part of XSL which allows us to identify parts of a particular XML document (elements, attributes).

XPointer:
: A language which allows us to identify any point of an XML document. It is an extension of XPath and is used in combination with XLink.

XQuery:
: It is a query language for XML, either on isolated XML documents, or on databases of XML documents.

XSL:
: eXtensible Stylesheet Language. Stylesheet language for XML, standardised by the W3C. It comprises 3 parts: XPath, XSLT and XSL-FO.

XSLT:
: XSL Transformations. Part of XSL which allows us to define how to transform an XML document into another document in XML format, HTML or plain text.

XSL-FO:
: XSL Formatting Objects. Part of XSL which allows us to define how to present an XML document.

XML: The ASCII of the 21st Century

Carlos Delgado-Kloos and Luis Sánchez-Fernández

In this article the authors aim to give readers an introduction to the world of XML. The article includes a review of its origins, a short introduction to the XML language and the main standards related to it, a look into the reasons behind its success and some brief conclusions.

Keywords: XML, SGML, Metalanguage, Metadata, DTD, World Wide Web.

1 A brief history

For the origins of XML we need to go back to the 80s when as a result of an initiative by publishing houses in the USA a language known as SGML (*Standard Generalized Markup Language*) was defined. SGML, which was the successor to another language, owned by IBM, called GML, was also useful to the publishing world. The idea was to develop a language in which it would be possible to separate the two most relevant functions in the publishing world, contents and the way these contents, in this case books and publications, could be presented. The author of a publication is the content specialist, however the publisher is the person who is best able to define how that content should be presented. SGML allowed specific markup languages to be defined, and thus can be considered as a metalanguage, a language or notation for defining languages. Thus SGML is a language which has nothing to do with the Internet or with networks.

When Tim Berners-Lee developed the *World Wide Web* in the late 80s he needed a language to express the contents which were to be circulated on the Net and presented via a browser. So he used this SGML metalanguage as a basis for defining the original HTML (*HyperText Markup Language*). HTML grew very quickly as additional requirements were demanded of it, but it also grew haphazardly as companies fought to maintain their leadership. More and more capabilities, more and more elements (tables, backgrounds, audio and video objects, etc.) were added, causing HTML to be gradually extended. But where was HTML's limit? Would HTML have to grow indefinitely? Different groups wanted to incorporate their own particular requirements. For example, mathematicians demanded to be able to express mathematical formulae correctly (with their indexes, sub-indexes, integers, fractions, etc.). But other groups had exactly the same rights: chemists wanted to represent chemical formulas while for musicians it was their musical scores they wished to be represented in a reasonable way. If all these different demands were to be met, HTML would become a hugely cumbersome language, and so at that moment it made no sense to meet all the different requirements.

People started thinking, right, let's go back to SGML. SGML was a metalanguage with which specific languages and applications could be defined for different uses. So instead of defining a markup language for multiple uses, each user group could be left to define their own markup language. But there was a problem: the high complexity of SGML. SGML had some very complicated manuals and had also been defined without bearing in mind current theory of formal languages and automatons: in other words it did not incorporate state of the art theory for language processors. It was very complex to define specific languages and it was very complicated to define processors for

Carlos Delgado-Kloos is presently a Full Professor at the Universidad Carlos III of Madrid (Spain), where he also holds the posts of Director of the Department of Telematics Engineering and Director of the Master in E-commerce. His current interests include design languages and techniques, as well as Internet-based applications such as electronic publishing, tele-education and e-commerce. He has been involved in many projects with European, national (Spanish Ministry) and bilateral (Spanish-German and Spanish-French) funding. He has published over 60 articles in national and international conferences and journals, and has also written one book and co-edited four more. He holds or has held various posts in national and international bodies such as: vice-president of IFIP TC 10, secretary of IFIP WG 10.5, editor of the Springer journal 'Formal Aspects of Computing', assistant director of Telecommunication Engineering at his University and manager of the National Programme for Information and Communication Technologies at the Spanish Ministry for Science and Research. He has been programme committee member or chair at more than 40 conferences and workshops, including vice-programme chair for the IFIP'92 World Computer Congress and programme chair for DATE 2002. He belongs to several professional associations in Spain and abroad, ATI being one of them, and has published several papers in Novática, ATI's journal whose Editorial Board he was a member of.
<cdk@it.uc3m.es>

Luis Sánchez-Fernández is a Full Professor in the Department of Telematics Engineering of the Universidad Carlos III of Madrid (Spain). He is a Doctor of Telecommunications Engineering from the Universidad Politécnica de Madrid. He is the author of more than 30 publications in journals and national and international conferences. He has taken part in several national and international research projects. Currently, his main interest is in the following fields: electronic publishing, electronic journalism and XML. He is member of ATI (Asociación de Técnicos de Informática, Spain.) <luis@it.uc3m.es>

those languages, which were also relatively inefficient. Which was why XML came along.

XML (*eXtensible Markup Language*), is a simplified, stripped down version of SGML, which may reduce freedom to define specific languages, but provides greater flexibility and makes it much simpler to define more efficient language processors. It is said that with 20% of the potential SGML had you can get 80% of the results.

How are these specific languages defined? If a mathematician wants to define his specific markup language – or a musician, chemist etc. decides to do the same – they have to use a notation. That notation, standardised by the W3C February 10, 1998 is called DTD (*Document Type Definition*), following the tradition established by SGML. A DTD defines the specific type of application which we want to use.

However in May 2001 the W3C set out another way of defining XML applications: XML Schema. This notation is a much more powerful, expressive and complete way of defining specific languages. There are therefore two notations for defining specific applications currently in existence, each with their advantages and drawbacks when it comes to defining new applications.

2 Trees

We are going to digress for a moment and quote from a book which has nothing to do with either the Internet or new technologies. It is an excerpt from the book *"Zen and the Art of Motorcycle Maintenance"* by Robert Pirsig:

"The structure of concepts is formally called a hierarchy *and since ancient times has been a basic structure for all western knowledge. Kingdoms, empires, churches, armies have all been structured into hierarchies. Tables of contents of reference material are so structured, mechanical assemblies, computer software, all scientific and technical knowledge is so structured..."*

In other words, in all these environments there is a head, a king, an emperor, a Pope, a general and then a hierarchical structure. Well that same hierarchy, which forms the basis of all spheres of western society, is also used in the context of markup languages. XML is no more than a language for defining hierarchies. XML allows us to represent hierarchical information, with us, the users, defining the elements of the hierarchy. The hierarchy is what in computer science is known as a "tree", which has a node with sub-nodes (sub-trees) hanging from it, all of these elements being recursively subordinate to it. If we want to represent the hierarchy textually or sequentially, rather than graphically, we make use of what we call parentheses, which are opened and closed to represent the beginning and end of each hierarchical element. Thus XML is nothing more than a simple language for representing hierarchical information or for representing user-defined parenthesis languages.

3 The XML family

XML is a *metalanguage*. By this we mean that XML permits the definition of different *markup languages* by using DTDs or Schemas. A markup language is a language which allows us to add marks or tags to describe a document's content in order to give semantic meaning or indicate how the content is to be presented. Generally speaking, the content is text.

Without wishing to go into a detailed description of what an XML document is like, it would be a good idea at this juncture to introduce the basic concepts involved. In XML, tags start with the symbol "<" and end with the symbol ">".

Generally speaking tags encapsulate a particular chunk of content which they are describing. In this case, the tag placed immediately before the content we want to encapsulate is written with a name enclosed by the symbols "<" and ">", for example <judge>. The tag placed immediately after the content begins with "</", for example </judge>. The name written in each tag must be the same. For example:

```
<judge>Don Francisco Ortega Ortega</judge>
```

The content encapsulated in a pair of tags may in turn contain other text tags:

```
<court>
    <province>Madrid</province>
    <town>Madrid</town>
</court>
```

In XML terminology, a pair of tags together with their content is called an *element*. There may be an element which does not contain anything. In that case the pair of tags can be replaced by one single tag ending in "/>", for example <acquitted/>.

The elements may contain attributes. An attribute is a pair (name, value) which gives particular information about an element. An attribute is written within the opening tag of the element, after which comes the name of the tag, using the syntax name="value". For example:

```
<judge promotion="1997">
    Don Francisco Ortega Ortega
</judge>
```

A correct XML document must comply with a series of rules. For example, the tags should be correctly nested. Therefore the following text would not be correct:

```
<court>
    <province>Madrid</court>
</province>
```

An XML document complying with this rule and others which we won't go into now is said to be "well formed".

A DTD defines what the XML documents used in a particular application domain are like. Among other things a DTD allows us to define:

- the tags which can be used
- the content of each element, that is, what elements can form part of a tag content and whether the element can contain text or not
- attributes of each element, optional or obligatory

The DTD can form part of an XML document, but the normal thing is for it to be in a separate document, so it can be referenced by the different XML documents which use that DTD.

When a well-formed XML document follows the rules defined in a DTD (the DTD which an XML document uses is indicated in the document itself), the XML document is said to be "valid".

There are a great many applications in which DTDs are used. For example, there are DTDs used in the fields of journalism (NewsML, NITF), law, e-commerce, to integrate IT tools, even in biotech applications.

DTDs have some limitations. For example, it is not possible to define that the text encapsulated within a pair of tags or the value of an attribute is a number. This would be useful on many occasions, for example to define a date, a quantity, etc.

Another of DTD's limitations concerns what can be specified with regard to the content of an element. For example, we can specify in a DTD that an element can contain another element 1 time, 0 to 1 times, 0 to several times or 1 to several times, but it is not possible to define (or at least not easily) that an element should contain another element between 3 and 15 times.

These and other limitations have caused a new standard to be developed, XML *Schema*, which allows us to define what XML documents used in a particular application are like more precisely than DTDs do.

In certain situations, it may be necessary to use elements defined in several DTDs. For example, an e-store may use one DTD for its purchase orders and another for its product catalogue. It may be desirable to keep the two DTDs separate for several reasons: use of standard DTDs or to keep different parts of the e-store separate. However, it is possible for a purchase order to include information defined in the catalogue DTD: name and product code number, price, etc.

Unlike HTML, the tags of an XML document say nothing about how the document should be seen on a browser, since they have been defined by the user. It is therefore necessary to provide XML with the means to define how to present an XML document. This is done by means of XSL and CSS stylesheets which we will be looking at later.

On the other hand, the fact that nothing is said in an XML document about the presentation gives us great flexibility when it comes to deciding how to present it. We can make several different presentations of the same XML document. We can define how we want an XML document to be seen on different platforms (Web browser, mobile phone, paper). We can decide which parts of the document we want to present and which parts we want to hide.

XSL (*eXtensible Stylesheet Language*) is a vital tool when it comes to managing XML documents. Its simplicity makes it the preferred means of managing XML documents by non-experts.

XSL allows us 1) to define how to present an XML document and 2) to convert an XML document into another document in XML format, HTML or plain text (which allows us to convert into any presentation format using plain text, such as RTF or PostScript).

As we mentioned earlier, one of the advantages of XML is that it allows us to separate the semantic meaning of a particular document from how the document's content is to be presented. The method used, practically exclusively, to describe how to present an XML document is by CSS stylesheets (which we take a look at later) and XSL stylesheets.

By using XSL stylesheets it is possible to convert an XML document to any presentation format (HTML, PostScript, PDF, WML for mobiles, etc.), which allows us to present the same XML document on different platforms: a computer with a Web browser, a printable version, another for a mobile phone, another for digital TV, etc. Naturally, it is possible to have a set of stylesheets which generate versions in different formats of the same XML document and it is even possible to have several stylesheets for the same format (for example to generate several HTML versions of the same XML document).

It is often necessary to carry out a particular treatment of XML documents. If this treatment is not excessively complex, it is possible to do it by using XSL stylesheets. For example, let us imagine an e-store which receives from a publisher XML documents describing the books published by the publisher which use the publisher's own proprietary DTD. The e-store in turn uses its own DTD for describing the catalogue of products (which includes other products besides books) it offers to its customers. In this situation it could be very useful to have an XSL stylesheet to convert the XML documents sent by publisher into other XML documents conforming to the DTD used by the e-store.

In the field we are concerned with, the problem of dumping could be solved by means of an attribute with the name "dumpable" and values "yes" or "no" which identify the parts of the text which need dumping. With an XSL stylesheet it is possible to generate the HTML, PDF, etc. document from which the parts to be dumped have been eliminated directly.

XSL is actually made up of three standards:

- XPath: is a notation which allows us to make a reference to part of an XML document. This can mean that reference is made to several chunks of the XML document. For example, in XPath we can make reference to all the elements of an XML document with the tag <magistrate>.
- XSLT (*XSL Transformations*): is a language which allows us to define transformations for converting an XML document into another document in XML format, HTML or plain text. XSLT documents are also XML documents. XSLT uses XPath.
- XSL-FO (*XSL Formatting Objects*): is an XML application whose semantic defines how to present an XML document which uses the components defined in XSL-FO. It allows us to describe the presentation of the document, defining things such as areas within the presentation space, size, position, colour, type of letter, etc.

CSS (*Cascading Style Sheets*) are an alternative to XSL for defining how to present an XML document. CSS allows us to define how to present an XML document on a browser or another device (for example, it allows us to define how to print out a document). Note that in the case of CSS stylesheets a new document is not produced (as is the case of XSL), but the stylesheet itself defines how to present a given XML document.

The presentation of an XML document using CSS is based on associating properties to the elements of the XML document which refer to how they should be presented: letter type, size, source, margins, etc.

Name spaces allow us to define an XML document which uses elements obtained from different sources. They do this by associating prefixes to each element tag name and each prefix is unmistakeably identified by means of a URI (*Uniform Resource Identifier*), a generalization of the well known URLs.

The last two standards we are going to look at in this short presentation of XML are XLink and XPointer. These two standards jointly allow us to define links between XML documents. They play the role of hyperlinks in the HTML world, but they are much more powerful.

The standard which allows us to define the links is XLink. It differentiates between two types of links. The simple type allows a particular resource of an XML document to point to another resource. It is similar to HTML hyperlinks. A resource is defined as a unit of information or a service which it is possible to make a reference to. Therefore an XML document is a resource. A particular element of an XML document rather than the complete XML document is also a resource.

The extended type is the most general form of define links. In fact, the simple type is a special case of the extended type. With extended type links it is possible to define links in which there are several source resources which point to several target resources (unlike the simple type in which one particular resource points to another).

When a link is defined, it is necessary to specify the source and target resources involved in the link (in the simple type links it is only necessary to specify the target resource, since the source resource is the element where the link is defined). Xpointer's role is to be able to identify a particular resource. It is really an extension of XPath. In XPath it is only possible make reference to elements or attributes. XPointer also allows us to identify a particular place in the text which has an XML element (with XPath only a complete element could be referenced).

4 XML features

XML (*eXtensible Markup Language*) meets the need to have formats for electronically describing different types of contents, in order to store, process and transmit them. Without going into specifically IT oriented applications, in a great many fields it is necessary to have notations which can describe contents.

By way of example, in the field of journalism there have been formats since the late 70s which allow the description of news. By using these formats, together with the news text, information can be included that does not strictly form part of the news text, such as the date, the section to which the news item belongs or to identify the part of the text which makes up the headline. These formats are used for transmitting the news from news agencies to a communication media or for storing the news on a database and where efficient searches can be subsequently made. The formats initially developed have become obsolete, and new formats using XML are currently being developed.

Software tools are using XML more and more. Below are just some of the many uses of XML in this area: XML is used as an internal format certain tools; it is also used to provide the user with documentation about the tool, and by using stylesheets it allows this documentation to appear in an HTML version to be seen on a browser, a PDF version for printing, and it can even allow the user to access the documentation in XML format directly in order to make searches on structured data. Another use which is becoming increasingly important is the use of XML as an information interchange format between different software tools which need to be interoperable.

XML is not the first technology to be developed with these aims in mind, nor will it be the last. We should therefore consider why we should use XML instead of one of the other alternatives on offer. Below we list just some of XML's advantages.

XML is powerful. By means of DTDs and XML Schemas, it is possible to define how we want the XML documents of a particular application to be. It is possible to make new versions of how we want the documents to be (that is, it is possible to modify the DTD or Schema) at a reasonable cost.

XML separates the aspects of content description from the aspects of presentation. In an XML document it is possible to include all the information needed to describe the content: document structure, cataloguing of the whole document and of its parts (metadata), definition of related documents (links), etc. In an XML based work environment the normal thing is for the XML document to describe the information which we want to add to the text of the document (as we saw before in the example concerning journalism), but not to include aspects of presentation (letter type, size, typesetting, etc.).

Later, by using stylesheet technology, it is possible to generate versions of the document in different formats, such as HTML (for publishing the document on the Web), plain text, PDF, etc. The fact that the aspects of presentation are not normally described in an XML document makes it possible to make several versions of the same document in a same format, but with a different appearance. For example, using different colours for the background and the letter of an HTML page, etc. But if we wish, it is also possible to include presentation aspects in an XML document. However if the markup of the XML document consists of text format elements only, we should consider using other technologies, such as HTML combined with CSS.

XML is simple. XML is a simplification of SGML (*Standard Generalized Markup Language*). SGML is a powerful format for describing markup languages, but it has not enjoyed much success due to its complexity. Which is why the W3C decided to develop a simplified version of SGML which would retain most of its capabilities but would be much simpler. The result is XML.

XML is an open standard. XML is a standard by the W3C (*World Wide Web Consortium*), the organisation which develops Web related standards. The W3C has developed a family of standards around XML to facilitate the use of XML. The fact that XML is a standard is advantageous for several reasons. First of all it means that any software developer can develop tools for it, and so the use of XML does not tie us to any one company's products. Plus the fact that the standards development process at W3C is in itself a guarantee of quality.

XML is ever more widely used, which is speeding up its development: comparison with past experiences, existence of professionals who are experts in these technologies, development of all kinds of tools around XML benefiting from the growing market for such tools. The fields of application are practically limitless, bounded only by the imagination. It is impressive to see how many interest groups who had problems of how to represent their information have chosen XML as their base format, for example: motoring, chemistry, mathematics, education, law, advertising, tourism, human resources, banking, news, etc. It is incredible how all these disparate sectors are betting on XML.

XML is easy to implement. It is much easier to process documents with XML than with other formats. The use of DTDs and Schemas for defining XML applications, together with the tools available for processing XML documents makes it easier to develop computer applications based on XML than with other formats.

XML is not only understandable by computers. An XML document can be understood by a human with a minimum knowledge of XML. In this way it differs from EDI, for example, which is a cryptic language, since at the time it was designed saving bits was a major consideration. XML based equivalent languages for e-commerce can be read directly by a person if necessary.

5 Conclusion

Let us briefly go over the aims and reasons behind XML. SGML and its successor XML were originally conceived for defining markup languages which would separate form from content. For the Web a single specific language like HTML was no use, since a single language would have to meet the needs of too many interest groups and satisfy too many conditions to be of any use, which is why it was necessary to provide an agile framework which would allow new languages to be defined as they became necessary.

However XML's success did not come the way its designers intended. It was not a success for having separated form and content, but for a reason that had not been considered, which was that data has to circulate on the Net. This means that the best way to go is to structure those data hierarchically, and XML is perfectly suited to representing information hierarchically. Thus, although the original idea referred to presentation (and is therefore more a B2C – *Business-to-Consumer* – concept), the XML's success lies in the fact that it is a format for structured information interchange (and, as such, lies more in the realm of B2B – *Business-to-Business*).

Let us close with a quote from Dan Connolly of the W3C consortium:

"XML by itself is just a simple text format; but together with all the ways it's being used to share structured information, it's a revolution that promises to make the Web a whole lot smarter."

XML per se is nothing more than a very simple text format. It is almost nothing, and precisely for being almost nothing and for being so flexible it is causing a revolution with far-reaching repercussions. XML is the ASCII of the 21st century.

Translated by Steve Turpin

References

SAX Home Page:
<http://www.saxproject.org/>

Wireless Markup Language version 2 Specification.
WAP Forum: <http://www.wapforum.org/what/technical.htm>

Cascading Style Sheets, level 1,
Recommendation, January 1999. World Wide Web Consortium:
<http://www.w3.org/TR/REC-CSS1>

Cascading Style Sheets, level 2,
Recommendation, May 1998. World Wide Web Consortium:
<http://www.w3.org/TR/REC-CSS2>

Extensible Markup Language (XML) 1.0
Recommendation, October 2000. World Wide Web Consortium:
<http://www.w3.org/TR/REC-xml>

Extensible Stylesheet Language (XSL) 1.0 Recommendation,
October 2001. World Wide Web Consortium:
<http://www.w3.org/TR/xsl/>

Namespaces in XML, Recommendation, January 1999.
World Wide Web Consortium:
<http://www.w3.org/TR/REC-xml-names/>.

HyperText Markup Language Home Page.
World Wide Web Consortium:
<http://www.w3.org/MarkUp/>

The World Wide Web Consortium (W3C) Home Page:
<http://www.w3.org/>

W3C Document Object Model (DOM).
World Wide Web Consortium:
<http://www.w3.org/DOM/>

XML Linking Language (XLink) Version 1.0,
Recommendation, June 2001. World Wide Web Consortium:
<http://www.w3.org/TR/xlink/>

XML Pointer Language (XPointer) Version 1.0,
Candidate Recommendation, September 2001:
<http://www.w3.org/TR/xptr/>

XML Schema Part 0:
Primer, Recommendation, May 2001. World Wide Web Consortium: <http://www.w3.org/TR/xmlschema-0/>

XML Schema Part 1:
Structures, Recommendation, May 2001. World Wide Web Consortium: <http://www.w3.org/TR/xmlschema-1/>

XML Schema Part 2:
Data Types, Recommendation, May 2001. World Wide Web Consortium: <http://www.w3.org/TR/xmlschema-2/>

XQuery 1.0:
An XML Query Language, Working Draft, April 2002. World Wide Web Consortium: <http://www.w3.org/TR/xquery/>

XML, the Development of New Business Applications and the Software Industry

Enrique Bertrand-López de Roda

The explanation for XML's swift expansion can be found, among others, in the availability of a set of software products which guarantee a profitable and efficient use of this standard. We will examine the impact XML has had on several key areas of the world of business applications: development tools and platforms, storage systems and middleware and intra/inter-company integration infrastructures. We will not only talk about the new processing capabilities incorporated into existing products: ultimately XML opens up a need for a new family within the basic software of base, specifically linked to the support of this standard.

Keywords: XML, Business Applications, Software Industry

1 Introduction

The use of XML (*eXtensible Markup Language*) in all kinds of applications has being growing constantly in recent months, a faithful reflection of its degree of consolidation in the business world. In a very short time XML has gone through all the inevitable phases typical of all new technologies, which are, according to the Gartner Group of consultants' model are; inflated expectations, relative disappointment and, finally, productive stability.

This is a success born of the coming together of a series of very diverse factors, among which we could include the intrinsic virtues of the standard as such, or the existence of a large number of agreements of a corporate nature to share vocabularies and interchange XML based electronic documents.

In this article I want to concentrate on another of the crucial factors which explain this consolidation process: the availability of a wide range of software products and platforms which make the implementation of XML based applications increasingly more productive and manageable.

On the one hand, the XML "phenomenon" has changed the way our traditional development tools and infrastructures have evolved (for example, with the inclusion of XML parsers and XSLT stylesheet engines in most application servers). On the other hand, it has caused new families of software products to appear (for example, the so called native XML databases or the XSLT – *XSL Transformations* – schema and stylesheet editors). We will be looking at all of these in the following sections.

2 The value of XML in applications development: XML software

When we talk about XML based applications nowadays we are actually referring to any type of business IT system which, in one way or another, to a greater or lesser extent, makes use of the advantages that the standard has to offer. The commitment from most software manufacturers demonstrates that XML has become a technological catalyst which, to continue with the chemical metaphor, while often it does not explicitly appear in the "reaction", without its presence the result would not have been possible in a reasonable time and at an acceptable cost.

The XML specification is accompanied by a plethora of sub-standards (XSLT, XPath, DOM, XQuery, etc.) which, from a developer's viewpoint, are as, if not more, important than the core. As a whole they provide us with an acceptable base for building systems which are more interoperable and easier to adapt and maintain in an internet world. It is not the universal panacea – we know that such a thing simply does not exist in the world of software engineering – but it does make for a good starting point.

So why do we use XML? Because nowadays any electronic document expressed in this way is easier to define, validate, process and transform than if it were in any other format. But sometimes we forget that these advantages do not lie so much in XML per se (which at the end of the day is only a (meta)language of a descriptive nature) as in the existence of all the related software components which are what really simplify our work on applications.

This initial premise regarding XML is supported by the fact that today there is a very wide range of these components in any of the programming languages in regular use (Java, VB,

Enrique Bertrand-López de Roda is Director of Technology in Software AG España, where he has worked since 1986. He is a graduate in Chemical Sciences from the Universidad de Valencia (Spain). He began his professional career in the Spanish consulting firm ERIA and in his 20 years of experience he has worked as a software engineer, project manager and consultant in the design and construction of a great many IT systems. Currently his main activity is focused on the development of integration and e-services solutions, with particular emphasis on architectures based on the XML standard. For several years he was associate professor in the Department of Computer Engineering of the Universidad Autónoma de Madrid (Spain). <ebertrand@softwareag.es>

C/C++, JavaScript, Perl, Cobol, Natural, etc), and on any of the operating systems to be found in the business world (OS/390, Unix, Linux, Windows, etc.).

If we had to make do with just the two indispensable ingredients needed to develop XML applications they would be:

- An XML processor or parser, which processes an XML document, by verifying (if necessary) that it complies with the restrictions expressed in the corresponding schema and exposing a set of standard APIs (DOM, SAX, JDOM) to the programmes which allow us to manage/extract their content.
- An XSLT transformation engine or stylesheet processor, which allows us to apply an XSLT document or stylesheet to an XML document and obtain a document in any of the possible output formats (XML, plain text, HTML, WML, etc.).

The current trend is to incorporate these basic functions and other similar ones into the integrated development environment applications (IDEs) of most manufacturers of base software: IBM, Sun, Microsoft, Oracle, Software AG, etc. It can also be seen how the most important suppliers of business management and planning solutions, such as SAP or Siebel, have the latest versions of embedded interface products for sending/receiving XML documents.

3 The impact of XML on base software tools, platforms and products

The fact that, in a relatively short space of time, nearly all the development or business solution environments have taken on the capability of generating and processing XML documents has been a step in the right direction: a necessary step, but not enough.

When we look at what kind of applications benefit from the use of XML, new demands are emerging in terms of infrastructure and support tools. To fully understand these demands we need to take a look at the areas where we can speak with a degree of accuracy about XML applications.

In general, XML makes more sense in a system the more dependent the system is on the integrated concept of a complex electronic document which is divorced from any consideration regarding platform, both from the point of view of the producer and the consumer. This is a good moment to remember that this also means that there are applications in which it makes no sense to use XML.

It is therefore content management applications with multi-device access (corporate portals, document systems, etc.) and applications devoted to the integration and interchange of information between companies (systems B2Bi, marketplaces, etc.) which up to now have most benefited from XML's appearance on the scene. However, this is still a very limited number of uses in comparison to the new development model of web services applications.

As we all know, the web services model proposes a new IT architecture based on three basic ideas: firstly, the functions and processes of a company's business are encapsulated as components known as services which offer a simple and perfectly defined interface to the world outside the company; secondly, these services run in a context which allows access via web protocols web (mainly HTTP, but also SMTP or FTP); and finally, and most importantly for the purposes of our discussion, the services are described, located, invoked and responded in the form of XML documents (in accordance with the protocols and standards; SOAP, WSDL, UDDI, etc.).

Perhaps in the future many of a company's critical applications will be web services; i.e. XML applications in the strictest sense of the words. It is clear that in order to guarantee the productivity, viability, scalability, reliability, security, etc. of all these systems and of those we mentioned earlier, we will need somewhat more than just a development environment with a parser to meet our XML requirements.

Below are two clear examples of the first results in this respect:

- The emergence of a new type of tools that we might call XML IDEs. These integrate schema editors, XML document editors and XSLT stylesheet editors and their corresponding utilities for testing, evaluation of Xpath expressions, version control, etc.
- The appearance of global technological platforms for the development/exploitation of Internet applications, such as .NET from Microsoft or SunONE from Sun, fully based on and supported by XML and the web services model. This model is precisely the only thing they have in common: they use different languages, operating systems, and development standards, but they all share a commitment to XML.

Other results which are not so obvious and well known as the above arise from the inevitable demand for an acceptable, robust and reliable implementation of XML applications. Below are further examples of a clear demand for new types of software or the adaptation of the existing ones:

- The ability to store and publish a large number of XML documents with complex and/or mutable structures, which is especially relevant to the strategies of database manager manufacturers.
- The ease with which legacy corporate systems, which were obviously not designed for the use of XML, can be integrated, by means of this new type of applications. In this case there is a direct effect on the market for middleware and integration products.

Let us now look in greater detail at the impact which the XML standard has had on each of these two broad areas: storage and publication systems and middleware and integration infrastructures.

4 The storage and publication of XML documents

All the current debate regarding the options we have for storing XML documents hinges on one basic question: is it really necessary to store the XML documents which we receive or publish?

If we consider an XML document solely and exclusively as a temporary support to facilitate the interchange of information between systems, which can always be constructed (or decomposed) when necessary from the company's corporate data, the answer to the previous question is no. On the other hand, the answer is yes if we consider an XML document as an IT element of the business in its own right, which needs to be kept (for example, for legal or auditing purposes).

Once we rule out the most obvious and rudimentary possibility – that of storing XML documents as individual elements in a file system – for reasons of scalability, ease of search, security, etc. software manufacturers basically offer us two alternatives: either to use a relational database management system (RDBMS) with extensions to provide an XML "view" of the data stored or to use a new family of management systems known generically as "native" XML (XDBMS or NXD).

Supporters of the RDBMS with extensions option argue that really most of a company's critical data can be included in this type of products. So nothing could be simpler than to include an automatic XML/relational converter so that when someone requests an XML document it is constructed on the fly from all the tables containing the necessary data. In the opposite situation, storage, the inverse process is carried out and the document can be decomposed into tables (*mapping*).

If we are talking about very simple XML documents then clearly this can be a good option. But in fact the nature of an XML document, a nested and ordered hierarchy of elements, which may or may not have a formal schema behind it, is in strong contrast to the tabular, unordered and perfectly defined view of data in an RDBMS system.

With a complex XML document, that is, the type that real world applications manage, this "masking" mechanism reveals its limitations. It is not as transparent for the developer, who needs to help the supposedly automatic converter to configure the many underlying JOIN operations for example. And it can mean an additional workload for the administrators if they want to guarantee the system's performance: behind a simple retrieval or storage operation on an XML document there may be literally dozens of tables being accessed, some of which will have been created expressly to support some aspects of the conversion.

However the most serious long term problem this option has is the document's loss of integrity. There is often a legal requirement, especially in the world of company-company (B2Bi) integration world, which stipulates that the company sending or receiving the XML document should save a copy of that document. The popularisation of the web services model for carrying out commercial transactions with legal implications can only boost this demand.

For these cases extended RDBMSs change the focus and instead of table mapping opt for saving the document as a block in a kind of special column which supports binary large or character large objects (BLOB or CLOB). The price we have to pay for this is a severe reduction in the possibilities of subsequent retrieval allowed by a purely XML-based solution.

It is in this scenario where the second option proposed by some software manufacturers (Software AG among them) makes sense: "native" XML or XDBMS database management systems.

Given the "terminological noise" we suffer from in any activity related to information technologies, there is no precise definition of what is to be understood by a native XML database management system, but there is a certain degree of consensus with regard to some characteristics:

- XML documents whatever their nature (with or without schema) are stored as they are, without any explicit management or transformation of their structure. The basic unit of storage from a logical point of view is the document and not the elements it contains.
- When an XML document is retrieved, the result is identical to what was sent to the management system for storage. In other words, from the application's point of view the integrity of content and structure of any document is guaranteed.
- Collections of documents are defined, maintained and accessed by means of the pertinent XML W3C standards: XML Schema, XPath, DOM, XSLT, etc.

The basic advantage of XDBMS is its capacity for storing and managing XML documents which are complex and highly varied in terms of structure in an integrated way which is totally transparent for the developers. Some management systems also include indexing engines which are very efficient for searching for and retrieving documents totally or partially, by exploiting all the "contextual" potential that XML tags and the Xpath expressions offer.

We can also speak with some degree of accuracy about another category of related software, "XML servers". Certain XDBMS products add some very interesting functions to their basic ones of storage and retrieval of XML documents: the possibility of storing binary objects; the possibility of showing the results after dynamically applying XSLT documents; various gateways for accessing RDBMS or external file systems online and combining this data with the documents that are stored in the management system; or the possibility of developing extensions with new functionalities.

The unquestionable advantages of XDBMS in certain XML applications does not in any way mean that current RDBMS will be gradually replaced, in the same way that XML's expansion does not mean that the relational model will be abandoned. But the opposite proposition is also true: the relational model and the RDBMS have no place in areas where XML and XDBMS are irreplaceable. We are talking about two different and necessary focuses, data and documents, which are based on technologically different and necessary solutions. And as proof of this we have the recent announcements from Microsoft, IBM and Oracle with regard to future native XML management systems to come from these manufacturers.

5 The role of "middleware" and integration infrastructures in XML applications

To communicate with other companies by using XML documents is not always easy if we bear in mind that most of the existing applications in a company are incapable of generating or understanding this type of documents. The data representations they manage may also be very different from the final XML documents.

A typical example of this situation occurs in the web services which encapsulate the access to existent modules, perhaps only invokable with an RPC type call. We have to convert the XML document received into a call understandable by the module and we must convert its response (typically a data area) into an outgoing XML document.

The historical way to solve the problems of interoperability between software components written in different languages, following different standards and running on different operating systems, has been based and is still based on the use of middleware products. They are tools which, via a common programming interface (API), mask from developers the transformations and conversions necessary for two components to communicate.

It no surprise that the manufacturers of middleware products have reacted to the aforementioned demands by incorporating new capabilities linked to XML such as the following for example:

- Transformation functions for converting an XML message to a format which is understandable by the target application, or vice versa.
- Wrapping functions for generating calls to modules from XML documents in any language.
- Extraction/aggregation/fragmentation functions for constructing XML documents dynamically from different traditional data sources.

As in the case of storage, we can also witness the parallel phenomenon of new types of middleware platforms, especially those emerging to control and organise interchanges between companies (known as B2B integration servers). These servers simplify and systemise the repetitive tasks of validation, transformation, distribution, delivery, etc. of XML documents received or sent in the course of a commercial transaction. They also tend to integrate with native XML storage systems for everything to do with the guarantee of document persistence and registration for legal reasons.

Contrary to the claim that XML and web services are responsible for the disappearance of middleware products as surplus to requirements, the truth is quite different: the use of XML in a company which already has a set of business applications is only technically and economically possible with the support of this type of products.

6 Conclusions

The adaptation of existing base software and the appearance of new families of products aimed at supporting the XML standard more efficiently and productively is both proof of its consolidation and of the underlying reason for its swift propagation.

XML’s influence is already easy to see in the vast range of platforms, tools and infrastructures, which together form a new architecture for the IT systems of the future based on the “web services” model.

This is certainly good news for the XML world, because no standardisation initiative can prosper in the medium term if the software solutions manufacturers do not back it and make it viable.

Translated by Steve Turpin

Application of XML Mark-up Languages to Software Development

Baltasar Fernández-Manjón, Alfredo Fernández-Valmayor, Antonio Navarro, and José Luis Sierra

This paper outlines an approach to XML-based software development. According to this method, applications are described using domain specific, XML based, markup languages. With these languages we structure a set of XML documents that are subsequently processed to yield the executable application. The approach also makes an explicit distinction between contents documents and documents describing other application aspects (e.g. interaction, presentation and process). Using a software process model based on markup languages and documents we obtain some benefits such as an important code reuse and a significant maintenance improvement. This paper describes our experiences applying this approach in the hypermedia domain and in the development of an application framework for supporting a broader range of information-based applications.

Keywords: Markup Languages, Domain Specific Languages, Software Development, Hypermedia, Hypermedia Model, XML Processing Model, XML

1 Introduction

XML (eXtensible Markup Language) [W3C-XML 2000] is a language (in fact, a meta-language) used to define markup languages. These markup languages are used to structure documents. Furthermore, *descriptive* markup languages are used to establish the logic informational structure of a document independently of its processing. This structure is represented by a tree of elements. In these elements we can include a set of attributes that specify non-hierarchical information. XML defines these kind of trees using a grammatical formalism (DTD (Document Type Definition) or XML *Schema* [W3C-XML Schema 2001]. XML is not concerned with the processing of a XML-marked document. This aspect is solved in a different stage using the appropriate interpreter of the descriptive markup (in that manner the same document can be processed in different ways by different interpreters).

Initially, descriptive markup languages were the basis for applications designed to process very large repositories of documents. The main requirements of these type of applications were to provide a characterization of documents independent of: i) the actual content of every document; ii) the processing of the documents; and iii) the concrete platform. SGML (Standard Generalized Markup Language) [Goldfard 1990] predecessor of XML, was formulated to standardize the languages used in areas of application such as avionics or defence systems. In these applications the three former requirements of use are a primary goal. The usefulness of standard markup languages in other areas of application was soon demonstrated and new standards or extensions to previously established standards were developed. The most influential of these new languages was HyTime [ISO-HyTime 1997]. HyTime was developed as an SGML extension to facilitate the description of hypermedia applications. HyTime demonstrated that marked up documents could be used to describe the complexities of a non-trivial software application. In addition, other languages such as DSSSL (Document Style Semantics and Specification Language) [ISO 1996] were provided to deal with the processing of SGML documents. Most of these early initiatives have been reinterpreted in the XML world. The most analogous to HyTime are several XML-based languages that cover different aspects of a hypermedia application. SMIL (Synchronized Multimedia Integration Language) [W3C-

Baltasar Fernández-Manjón, MsD. and PhD. in Physics, is an Associated Professor in the Computer Systems and Programming (SIP) Department at the Universidad Complutense de Madrid (Spain). His main research interests are: educational uses of mark-up technologies, application of educational standards and user modelling. <balta@sip.ucm.es>

Alfredo Fernández-Valmayor, MsD. and PhD. in Physics, is an Associated Professor in the SIP Department at the Universidad Complutense de Madrid (Spain). Currently, his research activity is centred on the education uses of mark-up languages and in the development and authoring of educational materials for WWW educational systems. <alfredo@sip.ucm.es>

Antonio Navarro, MsD. and PhD. in Mathematics, is an Assistant Professor in the SIP Department at the Universidad Complutense de Madrid (Spain) He has recently finished his PhD. dissertation on the conceptualization, development process and prototyping of hypermedia applications. His main research interests are hypermedia development and hypermedia engineering. < anavarro@sip.ucm.es>

José-Luis Sierra, MsD. in Computer Science, is an Assistant Professor in the SIP Department at the Universidad Complutense de Madrid (Spain). He is currently working on a PhD. dissertation on adding operational semantics to mark-up languages as an approach to software development. His main research interests are focused on the processing of mark-up languages and their use in the development of applications based on domain-specific languages. <jlsierra@sip.ucm.es>

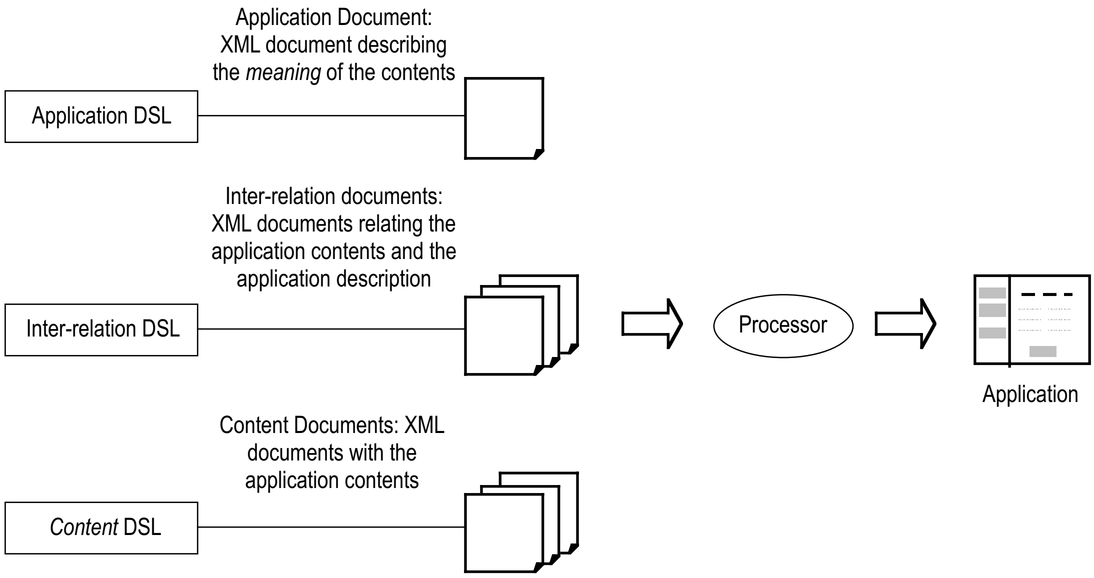


Figure 1: Using XML in a DSL-Based Approach to Software Development.

SMIL 2001] is a language or *application* [1] that facilitates the specification of the timing and synchronization in a multimedia presentation. XLink/Xpointer [W3C-XLink 2001] [W3C-XPointer 2001] are linguistic patterns used to describe different types of hyperlinks and anchors between documents. XSL (eXtensible Stylesheet Language), the XML-reinterpretation of DSSSL, [W3C-XSL 2001] [W3C-XPath 1999] [W3C-XLST 1999] introduces the notion of style sheet formed by a set of transformation rules that specify how to transform XML documents into other XML documents

The lack of expressiveness of existing languages, such as HTML, in a high-level characterization of educational hypermedia applications developed by our group in the last decade [Fernández-Manjón et al. 1997] motivated our interest in SGML. Using Dexter model precepts [Halasz/Schwartz 1994], we proposed a characterization of hypermedia applications based on a separation of the contents and links from the navigational schema (user interface plus navigational relationships). To represent both aspects of the hypermedia application (content structure and navigational schema) we used two SGML DTDs. The first one (*the content DTD*), specific to each application, describes the contents and its structure. The second one (*the presentation DTD*), generic to a family of multimedia presentations, characterizes the navigational schema of the application. Today, this idea has evolved into an approach to software development based on the definition of domain specific markup languages and the use of XML technologies. The applications suitable for being developed using this approach are described at high level using XML documents that are processed in subsequent stages of the development process to yield an executable application. In this paper we describe two of our experiences using this approach: the development of hypermedia applications, and the construction of a framework to support the development of a wider spectrum of content/information intensive applications.

1. In the XML context, *application* means a XML-based markup language.

2 Application Development Using Domain Specific Markup Languages.

We think that the use of descriptive markup languages to specify and to develop applications can be considered as a particular case of the approach to software development based on DSLs (Domain Specific Languages) [Deursen et al. 2000]. A DSL is a language specialized in the description of a type of problems and applications. These DSLs must provide a high degree of abstraction together with a set of language constructions (or syntactic categories) specifically adapted to the description of the problems at hand. These features of DSLs will furnish the development process only if the defined languages succeeded to capture the main aspects of the application domain.

XML technologies are at the core of our approach to application development. DSLs are defined using the grammar formalisms provided by XML and they must be able to express the main aspects of our application. This way, and after a first phase of domain analysis, we must obtain three different types of languages:

- One or more *content* DSLs able to express the internal structure of both the information and the data of our application.
- One *application* DSL able to describe in a declarative way the most relevant aspects of both the interface and the processing of the data.
- One inter-relation DSL able to relate content and application documents.

Using XML to define content and application languages makes it possible to express the relationship between content

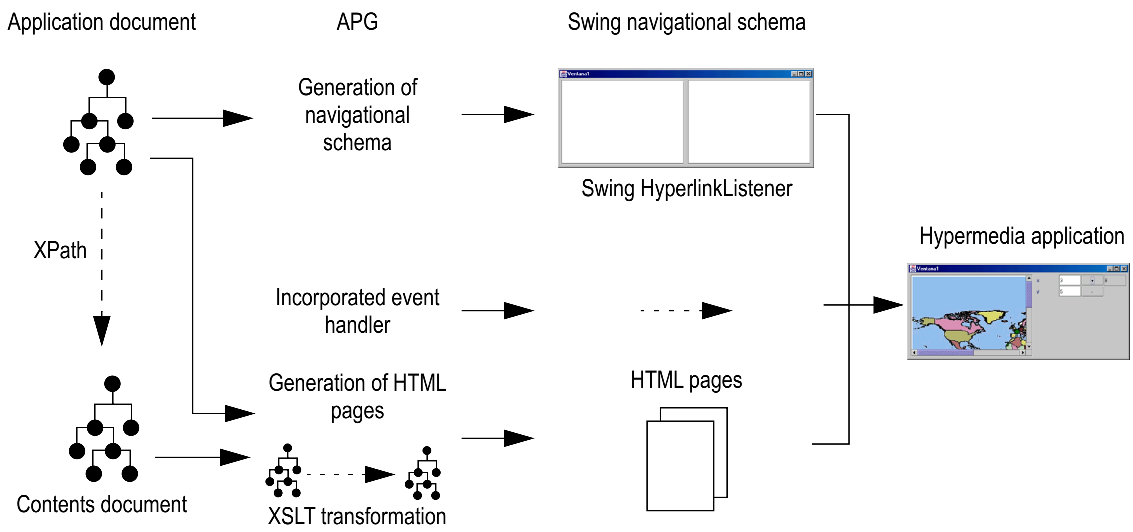


Figure 2: APG Working Scheme

and application documents using XML standardized languages such Xpath [W3C-XPath 1999] or XSLT [W3C-XLST 1999].

The construction of a specific application (Figure 1) implies to supply specific XML documents valid according to each of the DSLs. All these documents will be processed to obtain the executable application.

3 Development of Hypermedia Applications: Pipe and APG.

The main area of application where we have tested our approach has been the development of hypermedia software [Navarro et al. 2001]. Starting with the classic hypermedia models and making a detailed domain analysis (specifically of hypermedia in the educational domain) we have provided a high level characterization of hypermedia applications by means of an abstract model called *Pipe*. This hypermedia model characterizes applications using five components. The first component is the *contents graph* that is capable of representing, at the desired granularity level, the structure of the contents of the hypermedia application (using semantically defined nodes and links). The second is the *navigational schema* that provides the means to characterize the user interface and the navigational relationships between elements of the interface. The third is the set of *canalization functions*. These functions provide a means to relate the contents graph to the navigational schema. The main concept here is the concept of *canalization* (or *interpretation*) of the links defined at the content (semantically based) level through the links defined at the navigational level (or *pipes*), that is, the links connecting different elements of the interface. The fourth element of our model is the *navigational semantics*, that characterize the response of the hypermedia application when the user selects a link. Finally, there are the *presentation semantics* that describe the most specific characteristics of the application such as font size, family and colour.

Once we have built a Pipe representation of our application, we use two XML DTDs that, guided by the abstract model, serve to the generation of prototypes of the final application. The *content DTD* is specific to each application, and its purpose is to organize the content data of the application into a common tree structure that also makes explicit the relationships between the elements or nodes of the tree. The *application DTD* is common to all the applications and its purpose is to capture the navigational schema imposed over the contents. The relation between both instances of former DTDs (*contents document* and *application document*) is established using the *over-markup* technique. This technique uses XPath expressions on the application document to select the specific content elements (of the contents document) to be shown on each element of the interface, and it is an evolution of our former technique presented at [Navarro et al. 2000]. Notice that in this case, the content DSL is defined by means of the content DTD and that application DSL is defined by the application DTD. Finally XPath plays the role of the inter-relation DSL.

A Java program called APG (*Automatic Prototypes Generator*) produces a prototype of the final version of the application using content and application document. The current version of APG does not produce the final application because the process previously described only covers the conceptualization and prototyping phases of development [Navarro et al. 2002]. We are considering extending APG to cover the whole process of hypermedia development following an approach close to the one described by the Amsterdam model [Hardman 1994] and the language SMIL. Figure 2 sketches how APG works. Figure 3 shows a fragment of the application document and of the generated prototype.


```
<!DOCTYPE application SYSTEM “application.dtd”>
<application id= “tutorialesXML”>
  <window id=“v1”>
    <name>window 1</name>
      <pane id=“pl.1” size=”140, 320”>
        <paneContent>
          <defaultContent>/contents/index</defaultContent>
        </paneContent>
        <linksDestination pane=“pl.2”/>
      </pane>
      <pane id=“pl.2” size=“320, 200”>
        <paneContent>
          <groupContent>/content/subIndexes</groupContent>
        </paneContent>
        <linksDestination pane=“pl.3”/>
      </pane>
      <pane id= “pl.3” size=“400, 500”>
        <paneContent>
          <groupContent>/contents</groupContent>
        </paneContent>
      </pane>
  </window>
</application>
```

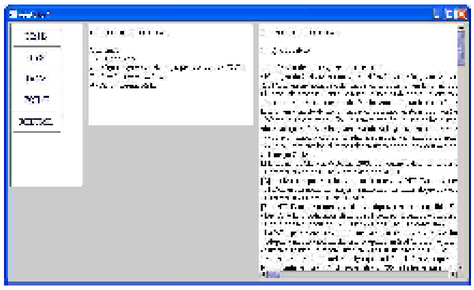


Figure 3: Application Document and Prototype

4 A Generic Framework for Application Development: DTC

In parallel to Pipe and APG approach, we have been working in the construction of a generic framework for application development following the guidelines sketched in Figure 1. To build this framework, instead of starting with a specific model like *Pipe*, we try to establish a more fundamental connection between the DSLs used to describe the specific features of an application and the markup languages used to implement these DSLs. Specifically, we focus on the problem of adding operational semantics to the XML languages that implement the DSLs [Sierra et al. 2000] [Sierra et al. 2001]. We call this approach DTC (Structured Documents, Document Transformations and Software Components).

DTC uses XSLT as the inter-relation language and associates a software component with each of the elements of the DTDs

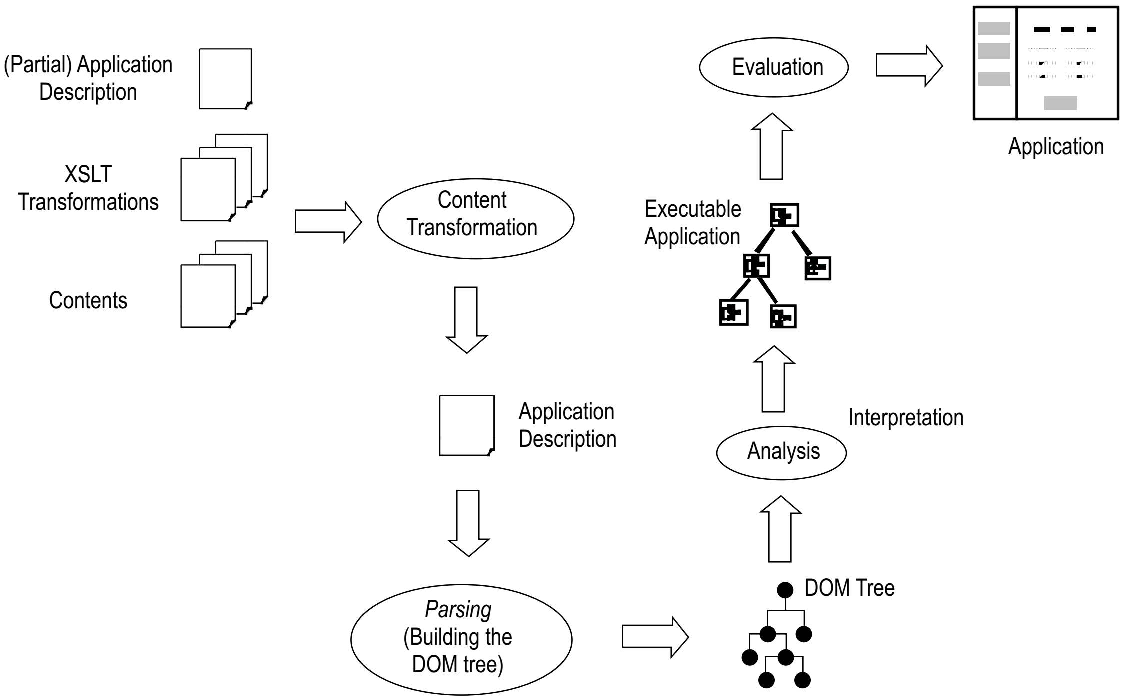


Figure 4: Generation of a DTC Application according to an *Analysis* and *Evaluation* Strategy.

that define each of the DSLs that describe the application. To build an application using the DTC approach we must provide: a) the contents documents, b) the XSLT documents needed to transform contents documents into application documents, and c) the rest of the application documents needed to fully describe the application and that could not have been generated by the transformations in b).

All the XML based application documents, generated in the first phase, can be integrated into a unique document using new XSLT transformations. From this final document we obtain the executable application: we first obtain the DOM (Document Object Model) [W3C-DOM 2000] tree of the document using a standard XML parser. Second, the DOM tree is processed by an extensible interpreter that, using the software components associated with each type of element, assembles the final applications as a hierarchy of objects. Third, to run the application we must activate the object at the root of the hierarchy of objects. This approach follows a well-known technique of construction of interpreters known as *analysis and evaluation* [Abelson/Sussman 1996]. The *analysis* phase corresponds to the processing of the DOM tree to assemble the hierarchy of objects. The *evaluation* phase corresponds to the execution or activation of the object at the root. Figure 4 sketches the process.

In DTC, the application DSL is formulated combining simpler languages (that is, combining new syntactic categories or elements), and this combination is made operational by associating an interpreter to each element. In order to facilitate the combination of interpreters, we included the possibility of adapting components by means of *interpreter adapters*. The resulting mechanism is similar to techniques for the construction of modular interpreters, like that described in [Liang et al. 1995].

At the moment we are working to provide a more precise characterization of the type of applications for which DTC could be advantageous, as well as to provide the CASE tools necessary in order to facilitate its application. Figure 5 schematizes the use of DTC to build an application that searches for the minimum path in a subway network. The structure of the subway network, together with its timetable and layout information (the latter used for the graphic presentation of the network) is described using the content DSLs. The DSL of the applica-

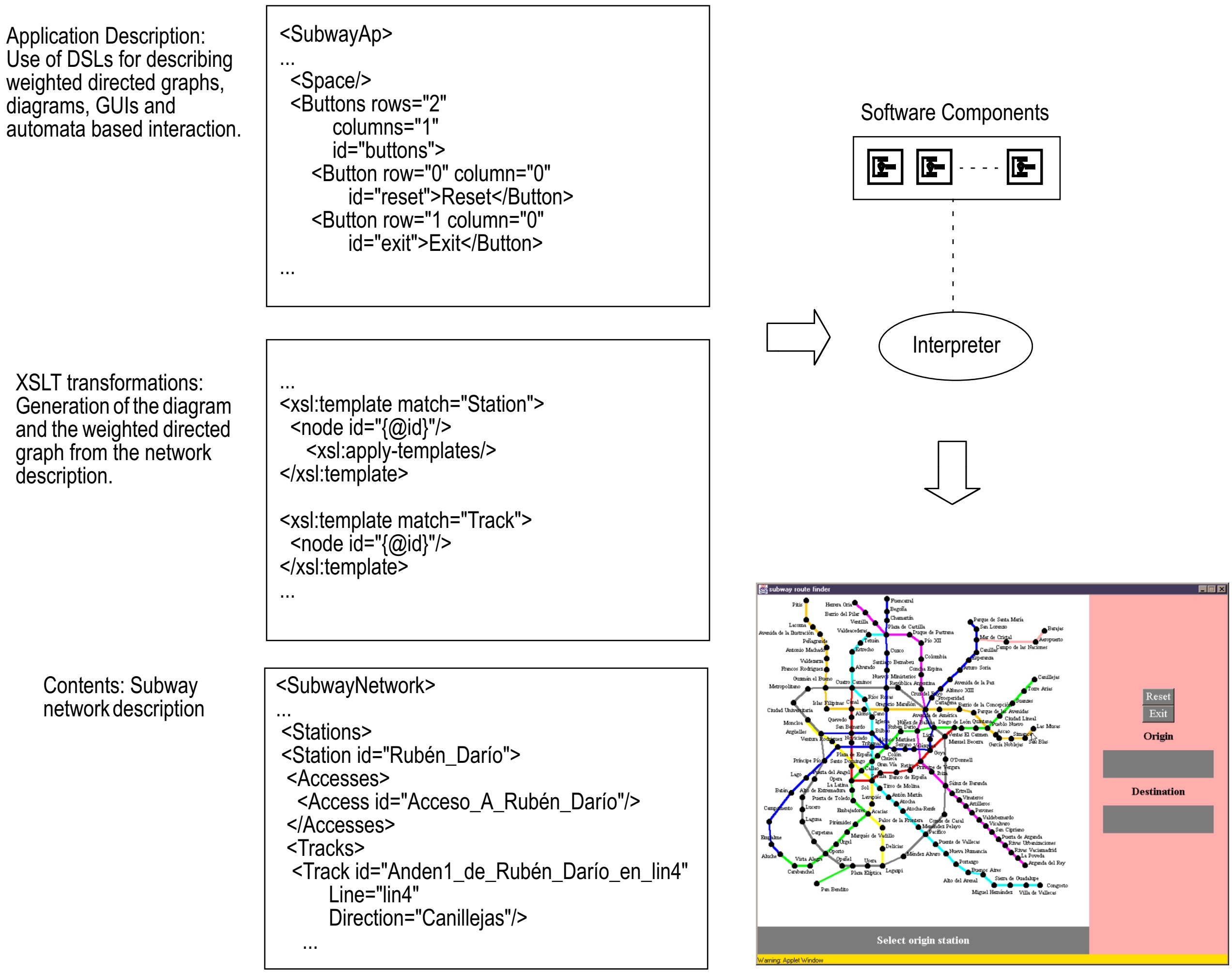


Figure 5: Subway Route Finder Application Generated using DTC

tion arises from the combination of a language to represent diagrams, a language to represent weighted directed graphs, a language to describe graphic user interfaces and a language to describe interactions (following a model of interaction based on automata). The descriptions of the diagram and of the graph are obtained by transforming the contents documents using XSLT transformations.

5 Conclusions and Future Work

We believe that markup languages and the associated XML technologies constitute a powerful approach for the construction of applications that, like in educational hypermedia, are based on an intensive use of different kinds of information. XML provides readability and declarativeness, and also provides means to explicitly describe the structure of the information characteristics of a specific domain. Also, upon combining XML with programming languages like Java, we produce platform-independent applications portable to almost every environment.

On the other hand the separation of the different aspects that make up an application enables changes in one of them without affecting the other. This enables the rapid generation of prototypes and/or applications, simplifying the maintenance and the reuse of information. Once a domain has been characterized appropriately, we could provide specific markup languages for that domain, together with the interpreters of the languages. The final application is the result of the interpretation of the marked up documents that describe the whole application. The utilization of operational techniques like DTC facilitates the construction of those interpreters by combining simpler ones.

At the moment we are studying how to extend APG in order to support all the expressive power of Pipe. We are also analysing the possibility of combining these techniques with other models and formalisms of information representation (as the relational model) and their effective incorporation into the declarative representation of the application. The subsequent step is to increase the depth of extensibility and modularity that DTC offers. Our goal is to build modular interpreters for the DSLs of each type of applications. Likewise, we expect to carry out a more exhaustive characterization of the type of applications for which the DTC approach could be competitive, as well as an analysis of the CASE tools necessary in order to facilitate their application in an industrial development environment.

Acknowledgements

This work has been partially funded by the Spanish Commission of Science and Technology (CICYT) through the projects TIC2000-0737-C03-01 and TIC2001-1462.

References

[Abelson/Sussman 1996]
H. Abelson, G. J. Sussman: The Structure and Interpretation of Computer Programs. Second Edition. McGraw Hill. 1996.

[Deursen et al. 2000]
A. Deursen, P. van Klint, J. Visser: Domain Specific Languages: An Annotated Bibliography. ACM SIGPLAN Notices 35(6), pp 26–36. 2000.

[Fernández-Manjón et al. 1997]
B. Fernández-Manjón, A. Fernández-Valmayor, A. Navarro: Extending Web educational applications via SGML structuring and content-based capabilities. F. Verdejo y G. Davies (eds.), The Virtual Campus: Trends for Higher Education and Training, pp 244–259. Chapman & Hall, Londres, 1997.

[Goldfard 1990]
C. F. Goldfard: The SGML Handbook. Oxford University Press, 1990. Oxford University Press,1990.

[Halasz/Schwartz 1994]
F. Halasz, M. Schwartz: The Dexter Hypertext Reference Model. Communications of the ACM 37 (2), 30–39, 1994.

[Hardman 1994]
L. Hardman, D. C. A. Bulterman, G. van Rossum: The Amsterdam Hypermedia Model: Adding Time and Context to the Dexter Model. Communications of the ACM 37 (2), 50–62, 1994.

[ISO 1996]
International Organization for Standardization. DSSSL – Document Style Semantics and Specification Language. ISO/IEC 10179. 1996.

[ISO-HyTime 1997]
International Organization for Standardization. Hypermedia/Time-based Structuring Language (HyTime) – 2d Edition. ISO/IEC 10744, 1997.

[Liang et al. 1995]
S. Liang, P. Hudak, M. P. Jones: Monad Transformers and Modular Interpreters. 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages. San Francisco, CA. 1995.

[Navarro et al. 2000]
A. Navarro, B. Fernández-Manjón, A. Fernández-Valmayor, J. L. Sierra: A Practical Methodology for the Development of Educational Hypermedias. International Conference on Educational Uses of Information and Communication Technologies, ICEUT 2000. 16th IFIP World Computer Congress 2000, Information Processing Beyond Year 2000, pp 217–220. Beijing, 2000.

[Navarro et al. 2002]
A. Navarro, B. Fernández-Manjón, A. Fernández-Valmayor, J. L. Sierra: Formal-Driven Conceptualization and Prototyping of Hypermedia Applications. Fundamentals Approaches to Software Engineering 2002, FASE 2002. European joint conferences on Theory and Practice of Software 2002, ETAPS 2002, pp. 308–322. Grenoble, 6-14 april, 2002. Published as Lecture Notes in Computer Science (Vol. 2306), Springer-Verlag, Berlin, 2002.

[Navarro et al. 2001]
A. Navarro, A. Fernández-Valmayor, B. Fernández-Manjón, J. L. Sierra: Using Analysis, Design and Development of Hypermedia Applications in the Educational Domain. M. Ortega y J. Bravo (Eds), Computers and Education: Towards an Interconnected Society, pp 251–260. Kluwer Academic Publisher, Holanda, 2001.

[Sierra et al. 2000]
J. L. Sierra, B. Fernández-Manjón, A. Fernández-Valmayor, A. Navarro: Integration of Markup Languages, Document Transformations and Software Components in the Development of Applications: The DTC Approach. International Conference on Software: Theory and Practice, ICS'2000. 16th IFIP World Computer Congress 2000, Information Processing Beyond Year 2000, pp 217–220. Beijing, 2000.

[Sierra et al. 2001]
J. L. Sierra, A. Fernández-Valmayor, B. Fernández-Manjón, A. Navarro: Operationalizing Application Descriptions with DTC: Building Applications With Generalized Markup Technologies. 13th International Conference on Software Engineering and Knowledge Engineering SEKE'01, pp 379–386. Buenos Aires. 2001.

[W3C-DOM 2000]
World Wide Web Consortium. Document Object Model (DOM) Level 2 (several volumes).
<http://www.w3c.org/DOM/DOMTR>, 2000.
[W3C-XML 2000]
World Wide Web Consortium. Extensible Markup Language (XML) 1.0 (Second Edition).
<http://www.w3.org/TR/REC-xml>, 2000.
[W3C-XSL 2001]
World Wide Web Consortium. Extensible Stylesheet Language (XSL) Version 1.0. <http://www.w3.org/TR/xsl/>, 2001.
[W3C-SMIL 2001]
World Wide Web Consortium. Synchronized Multimedia Integration Language (SMIL 2.0).
<http://www.w3.org/TR/smil20/>, 2001.
[W3C-XLink 2001]
World Wide Web Consortium. XML Linking Language (XLink) Version 1.0. <http://www.w3.org/TR/xlink/>, 2001.
[W3C-XPath 1999]
World Wide Web Consortium. XML Path Language (XPath) Version 1.0. <http://www.w3c.org/TR/xpath>, 1999.
[W3C-XPointer 2001]
World Wide Web Consortium. XML Pointer Language (XPointer) Version 1.0. <http://www.w3c.org/TR/xptr/>, 2001.
[W3C-XML Schema 2001]
World Wide Web Consortium. XML Schema Parts 0,1,2.
<http://www.w3c.org/XML/Schema>, 2001.
[W3C-XLST 1999]
World Wide Web Consortium. XSL Transformations (XSLT) Version 1.0. <http://www.w3.org/TR/xslt>, 1999.

Querying the Semantic Web: Feasibility Issues

José-Francisco Aldana-Montes, Antonio-César Gómez-Lora, Nathalie Moreno-Vergara, and María del Mar Roldán-García

At the moment, a whole new technology is being developed based on XML standards for the query processing of heterogeneous data sources. This includes, among other things, the specification of query languages and techniques for the mediation and integration of data sources. The next great challenge consists of developing what is already known as the Semantic Web, taking this technology as a starting point. It is the very efficient treatment of information, held in the logic and ontological layers, which is envisaged as one of the main factors that will determine its practical success.

Keywords: Data Integration, Processing Queries Based on Ontologies, Web, Semantics.

1 Introduction

Since the moment the Internet became the global network of communications and a medium for the development of information systems, it was evident that something more than static HTML pages was necessary. After the arrival of XML as a way to describe structured and semi-structured data, the World Wide Web Consortium (W3C) has been developing technology around this new standard. We can highlight XMLS (Extensible Mark-up Language Schemas), RDF (Resource Description Framework), RDF-Schema, XSL/XSLT (Extensible Style sheet Language/Transformation) and SOAP (Simple Object Access Protocol). These recommendations, together with the use of XML as the format for the electronic exchange of data, open the possibility of integrating heterogeneous data sources in the World Wide Web environment. As a result of this, the most important firms in software development, such as Oracle, Microsoft and IBM, have introduced support for XML in their databases and products.

All this technology has contributed to the development of new kinds of applications, which do not require human interaction, making the Web more "understandable" by machines. Berners-Lee, the creator of the Web, describes the Semantic Web as the next step in the evolution of Web applications, where "the information has a well defined meaning, making it possible for computers and people to work together" [Berners-Lee et al. 2001].

The development of the Semantic Web involves the adoption of different technologies (Figure 1), which allow the adding of meaning to the structure of XML documents and describing the necessary semantic information to enable processing by machines. For a long time this problem has been dealt with by means of "ontologies" described as documents or files "that absolutely define the relationships between terms". The ontologies allow working with concepts, instead of key words, in information retrieval systems. According to the information

José-Francisco Aldana-Montes received the Ph.D. degree from the University of Málaga (Spain) in 1998. He presently holds the rank of Assistant Professor in the Computer Sciences Department of the University of Málaga. He acted as program committee member for the JISBD from 1999–2002. His current research interests involves the study of integration of database and Web technologies. This includes (distributed) evaluation and optimization of recursive queries; distributed evaluation of XQuery; (semantic) optimization of XML queries; semantic integration and theoretical models for the Semantic Web; and query processing in the Semantic Web. <jfam@lcc.uma.es>

Antonio-César Gómez-Lora is a Teaching Assistant in the Department of Computer Sciences of the Universidad de Málaga (Spain). He received the M.S. degree in Computer Sciences from the Universidad de Málaga in 1997 and he is currently working towards the Ph.D. degree at this University. His research interests include hybrid techniques for recursive query evaluation and optimization in distributed systems and optimization techniques at query evaluation time. <cesar@lcc.uma.es>

Nathalie Moreno-Vergara is a PhD Student, supported by a Research Grant, in the Department of Computer Sciences of the Universidad de Málaga (Spain). She received the M.S. degree in Computer Sciences from the Universidad de Málaga in 2000 and she is currently working towards the Ph.D. degree at this University. Her research interests are focused on ontologies, semantic integration and theoretical models for the Semantic Web (Query languages and reasoning mechanism for that environment). <vergara@lcc.uma.es>

María del Mar Roldán-García is a PhD Student, supported by a Research Grant, in the Department of Computer Sciences of the Universidad de Málaga (Spain). She received the M.S. degree in Computer Sciences from the Universidad de Málaga in 2000 and she is currently working towards the Ph.D. degree at this University. Her research interests include heterogeneous data sources integration, ontologies and indexing techniques. <mmar@lcc.uma.es>

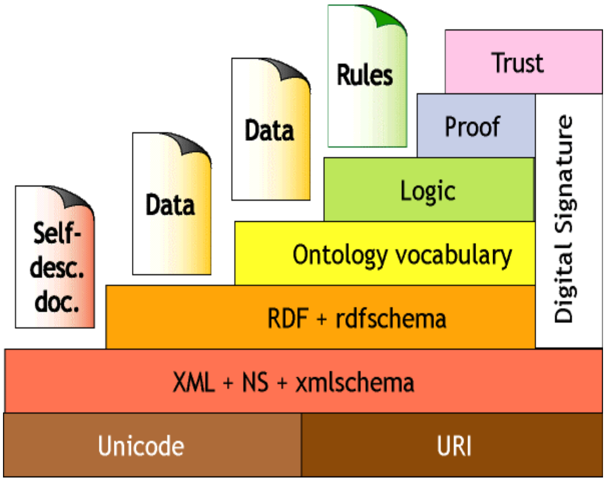


Figure 1: View of the Semantic Web presented by Berners-Lee

sources, the ontologies describe the content of data repositories regardless of their syntactic representation, enabling their semantic integration [Mena et al. 2000]. A problem that will arise in the near future, due to the rapid spreading of specific ontologies (and standards for metadata), is the integration of such ontologies.

2 Languages for the Semantic Web

In order to materialise a conceptualisation or ontology, a language to represent that knowledge is required. Traditionally, these languages have been developed in the area of Artificial Intelligence and focus their formal base on paradigms such as 1st and 2nd order predicate calculus, description logic or object oriented paradigms. It is the different computational and expressive properties that later differentiate these languages. Among the most outstanding ones are Ontolingua, Loom, OCML and Flogic.

Since XML became a "de facto" standard for the exchange of data between applications, languages of representation of knowledge for the Web were developed; these invite the publication of ontologies using the syntax of such a standard. This way, insofar as it is possible, the arduous task of having to define parsers is avoided. The languages based on XML are ideal candidates for the Semantic Web. Some of these are Simple HTML Ontology Extensions (SHOE), Ontology Exchange Language (XOL), Ontology Mark-up Language (OML) and Resource Description Framework Schema Language (RDFS). They inherit the labels of XML and incorporate new characteristics that improve the expressiveness of the initial data model. Further language proposals extend RDF and RDFS, such as: Ontology Interchange Language (OIL) and its successor DAML+OIL.

Given that XML imposes the necessity of structural restriction (a common core), in order to provide unmistakable methods of semantic expression, RDF just provides the infrastructure to allow the codification, reuse and exchange of structured metadata. In this way, RDF is the most promising model to associate information to the content of Web resources.

It is not likely that a single language will cover all the needs and requirements presented in the Semantic Web. An analysis of the most desired features, regarding expressiveness and power of reasoning mechanisms, will provide us with the profile of an efficient language for the Semantic Web. A clear distinction between the terms is made: representation of knowledge and reasoning. The formalisation of knowledge is carried out, in most cases, through the use of concepts, n-ary relationships, functions, procedures, instances, axioms, rules of production and through formal semantics. These parameters determine the expressiveness of the language.

Table 1 shows a comparison over these elements, where the symbol '+' shows a aspect supported by the language, the symbol '–' shows an unsupported aspect, '+/–' shows an aspect not directly supported (but which can be modelled with the resources provided by the language) and 'NA' refers to unsupported aspects (of little relevance).

It can be seen that the languages based on XML do not usually provide the possibility of defining functions, procedures and axioms, except some limited axiomatisation such as the deductive rules in the case of SHOE. They also lack formal semantics inherent to the language. This makes it difficult the implementation of efficient mechanisms of reasoning. In this sense, Ontolingua is perhaps the most expressive of the formalisms presented, although at present there is no inference machine which implements it.

Other interesting comparisons can be carried out according to the reasoning mechanisms that the languages allow. Table 2 includes these aspects. OIL has an automatic system of classification (desirable in the case of the ontologies), Flogic implements the handling of exceptions and both have complete and sound systems of inference. Compared to the languages based on XML, traditional languages support the implementation of procedures, the maintenance of restrictions and both top-down and bottom-up evaluation. Thus an important trade-off exists between completeness and expressiveness of inference mechanisms used and their efficiency. This makes the study and

	Ontolingua	OCML	LOOM	FLOGIC	XOL	SHOE	RDF(S)	OIL
Concepts	+	+	+	+	+	+	+	+
n-ary Relations	+	+	+	+/–	–	+	+	+
Functions	+	+	+	+/–	–	–	–	+
Procedures	+	+	+	–	–	–	–	–
Instances	+	+	+	+	+	+	+	NA
Axioms	+	+	+	+	–	–	–	NA
Rules of Production	+	+	+	–	–	–	–	NA
Formal Semantics	+	+	+	+	+	–	–	–

Table 1: Comparison of languages according to the elements to represent the field of knowledge, presented in [Corcho/Gómez 2000]

	ONTOLINGUA	OCML	LOOM	FLOGIC	XOL	SHOE	RDF(S)	OIL
Inference Mechanisms								
Correct	–	+	+	+	–	–	–	+
Complete	–	–	–	+	–	–	–	+
Classification								
Automatic Classification.	–	–	+	–	–	–	–	+
Exceptions								
Use of Exceptions	–	–	–	+	–	–	–	–
In heritance								
Monotonic	+	+	+	+	NA	+	NA	+
Non-Monotonic	+/–	+/–	+	+	NA	–	NA	–
Simple Inheritance	+	+	+	+	NA	+	+	+
Multiple Inheritance	+	+	+	+	NA	+	+	+
Procedures								
Implementation of Procedures	+	+	+	–	–	–	–	–
Restrictions								
Examination of Restrictions	+	+	+	+	–	–	–	–
Evaluation Model								
Top-Down	–	+	+	+	–	NA	–	–
Bottom-Up	–	+	+	+	–	NA	–	–

Table 2: Mechanisms of language reasoning, presented in [Corcho/Gómez 2000]

development of techniques for the distributed evaluation of logical programs in this context very interesting; techniques that provide support to the query processing based on ontologies, in the Semantic Web.

3 A Framework for the Semantic Web: Practical Implications

Heflin proposes a new formal framework for the representation of knowledge in the Semantic Web, based on ontologies [Heflin 2001]. We are interested in the application of that generic framework to the integration and querying of Web services. Although Heflin is widening the formal definition of ontology during its development, to simplify the explanation we just use one of his fist definitions. An ontology is defined as the 3-tuple O = <V, A, E>, where V corresponds to the vocabulary, A is the set of axioms and E is the set of ontologies which are extended by O. Then there are two functions associated with resources: C(r) that determines the set of ontologies with which the resource r commits; and K(r) that translates the information of resource r, introducing it in the formal theory in terms of the ontology vocabulary (modelling, in a certain way, the concept of wrapping).

In Figure 2 we can see, in a graphical way, an example in which both O_U and O_F extend O_G. Additionally we have the perspective of O_U (a basic concept in this framework) in Figure 2. In the model presented by Heflin, the queries are carried out on the ontological perspectives. The perspective PO_U defined on the ontology O_U is the union of: the axioms of the ontology O_U; the axioms of all the ontologies extended by O_U; the functions of knowledge of all the resources that commit with the ontology O_U; and the functions of knowledge of all the resources that commit with all the ontologies extended by O_U.

This formal framework can be developed in many ways depending on different models of implementation, each of which has important practical implications (from the point of view of viability and efficiency of implementation) and theoretical implications (even expressing the need for extending "particularising" the previous model).

The most simple setting for the Semantic Web would be the one in which all the data sources commit with a very limited set of well known, universally accepted ontologies. Furthermore, all the data sources would be universal and homogeneously accessible. Thus, each system would collect the axioms and resources of its perspective quickly and would process them. It is evident that this scenario is not very realistic as a basic architecture for the Semantic Web. Not everybody is expected to commit to a limited group of ontologies, nor is an isolated system expected to process a whole perspective that involves a great many resources. It is easier to imagine that there will be many ontologies and many systems in charge of evaluating specific ontologies (SEO_i). This way, the perspectives should be made in a co-operative way, thanks to the interaction of these agents or systems. In this context it is essential to respect the interoperability and autonomy of the SEO_i. This requires a certain isolation of different levels, which we will analyse in three different dimensions. Each one of these presents a particular set of problems that must be dealt with in relation to aspects such as intellectual property, heterogeneity and distribution.

From the point of view of intellectual property, a firm may want to isolate or protect the data sources and/or axioms that define the elements of the vocabulary of its ontology. Taking the three basic elements that make up the model of an ontology, namely vocabulary, axioms and the data sources modelled by the functions of knowledge K(r), only the first of these must necessarily be public (totally or partially).

The second dimension with which we work is heterogeneity, based on the two key factors formalism and the inference mechanism. The extension of an ontology needs to be feasible even when using two different formalisms to represent the axioms. In the same way, two systems based on ontologies can

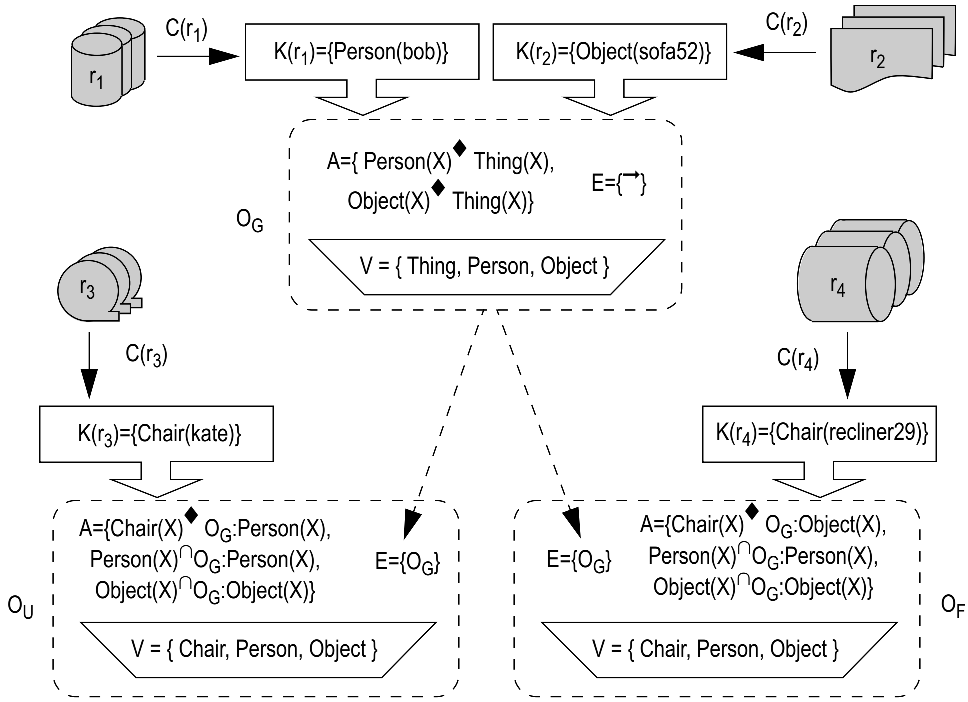


Figure 2: Example of ontologies, extracted from [Heflin 2001]

use different inference mechanisms to compute the facts of such axioms.

In the simple model of the implementation of Heflin's framework, we could think, perhaps, that the heterogeneity in the formalism could be solved introducing a translator. However, an extreme case of heterogeneity (that is both real and common) corresponds to the declaration of Value Added Services (VAS). A VAS in an integration or mediation system that can be interpreted as additionally functionality incorporated into system and whose semantics are not, and cannot be, expressed in the language used for its integration. The concept of VAS appears in many fields, such as, the stored procedures and external procedures in a relational DBMS or the external predicates in logical languages. In Heflin's framework, this would correspond to predicates of the ontology whose definition should not be carried out in the format used to express the set of axioms.

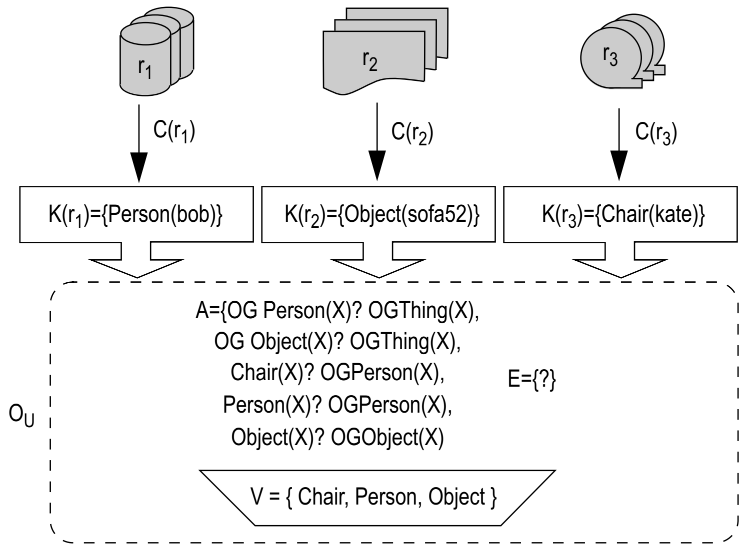


Figure 3: Formal theory equivalent to the perspective of O_U in the example of Figure 2.

The analysis of the third dimension, which we have called distribution, is aimed at considering aspects that are of paramount importance for the real viability of the Semantic Web. There are two basic aspects of distribution: the first refers to the evaluation of the ontology and the second refers to the optimization of such an evaluation. Regarding the evaluation, we run into important areas such as replication, load balancing and the level of tolerance of errors (errors attributable to loss "non accessibility" of data sources as well as loss of systems or agents of evaluation). Additionally, optimization techniques change radically when considering aspects of opacity regarding the resources, axioms, etc.

In summary, returning to the example in Figure 2, to maintain heterogeneity and autonomy we should obtain the same PO_U (Figure 3) in a context where the only known part is that shown in the diagram of Figure 4.

Keeping the original perspective of this diagram introduces new problems, since it requires that the systems in charge of evaluating O_U (SEO_U) and O_G (SEO_G) interact to get to the solution. This interac-

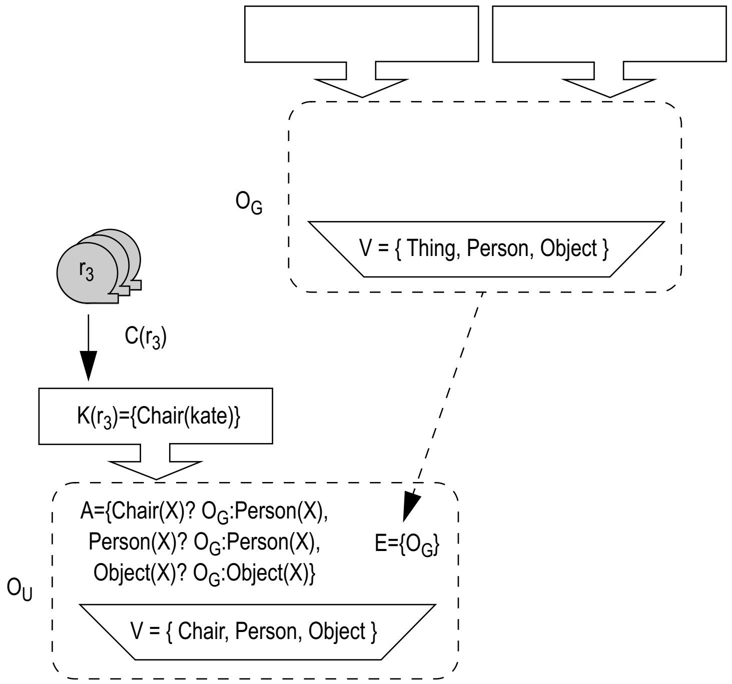


Figure 4: Opacity in the perspective of O_U.

tion is much more complex than the simple use of SEO_G by SEO_U. Both must co-operate to get to a valid solution. For instance, from PO_U it can be inferred OG:Person(kate), which at the same time will infer O_G:Thing(kate), see Figure 3. Thus SEO_U asks SEO_G whether Person(kate) is true. In a distributed framework (where SEO_U and SEO_G do not have all the information) SEO_G should answer, at the very best, "no" or "I don't know" (depending on whether it uses the hypothesis of a closed world or not).

From the point of view of constructing PO_U, without violating the autonomy of the ontologies or forcing a centred evaluation, if SEO_U informed SEO_G that Person(kate) is true in PO_U, rather than asking, then SEO_G will not only know this fact, but it will reply that Thing(kate) in PO_U is also true.

These interactions become more complex if we also take into account that, within a query, the answers belong to a Markovian model. In the previous example, this is equivalent to "recalling" the new facts Person(kate) and Thing(kate) in future interactions.

At the moment we are working on the further development of this formal framework, adapting and developing the techniques of databases to achieve a practical implementation of the model we have described. This is based on our previous work on the distributed evaluation and optimization of distributed logical programs by means of a model of data flow (D^3 – Distributed Dataflow Datalog) [Aldana 1998], [Aldana et al. 1996] and on the further development of mediating systems based on this model (D^3-Web) [Aldana et al. 1997], as well as on new work being developed in Data Stream Systems [Babcock et al. 2002].

4 Conclusions

The Semantic Web will continue to be a model of growth similar to the Web and its success depends on us being able to develop realistic techniques that make this development model feasible. The trade-off adopted between the concepts of power, completion and efficiency for each of the different inference mechanisms, will open a wide range of study for new evaluation techniques – based on ontologies – for distributed logical programmes within the context of the Semantic Web. However, the querying and efficient management of huge distributed knowledge bases still has many unsolved aspects, including those related to the efficient integration of information and the development of distributed inference mechanisms.

References

[Aldana et al. 1996]
J. F. Aldana, J.M.Troya, DataFlow Evaluation of Datalog Queries, en: International Joint Conference and Symposium on Logic Programming, Bonn, Alemania (1996) 69–78.

[Aldana et al. 1997]
J. F. Aldana, M.I.Yagüe, WWW as a Distributed Deductive Database, en: 8th International Conference on Management of Data, Madras, La India (1997) 277–292.

[Aldana 1998]
J. F. Aldana, Un Modelo de Flujo de Datos para la Evaluación de Consultas en Bases de Datos Deductivas. Tesis Doctoral. Universidad de Málaga. 1998.

[Babcock et al. 2002]
B. Babcock, S. Babu, M. Datar, R. Motwani, J. Widom, Models and Issues in Data Stream Systems. 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), (2002).

[Berners-Lee et al. 2001]
T. Berners-Lee, J. Hendler, O. Lassila. The Semantic Web. Scientific American. Mayo 2001.
<http://www.sciam.com/2001/0501issue/0501berners>.

[Corcho/Gómez 2000]
O. Corcho, A. Gómez. A Roadmap to Ontology specification Languages. Proceedings of Knowledge Acquisition, Modelling and Managements, 12th International Conference, EKAW 2000. Lecture Notes in Computer Science 1937 Springer 2000.

[Heflin 2001]
J. D. Heflin. Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment, PhD Thesis, 2001.

[Mena et al. 2000]
E. Mena, A. Illaramendi, V.Kashyap, A.P. Sheth "OBSERVER: An Approach for Query Processing in Global Information systems based on Interoperation Across Pre-Existing Ontologies". Distributed and Parallel Databases 8(2), 2000.

Digital Signature and Encryption with XML

Antonio F. Gómez-Skarmeta, María-Encarnación Martínez-González, Eduardo Martínez-Graciá, and Gregorio Martínez-Pérez

XML is useful for e-commerce applications because it uses standard tags to build business-like documents that can be automatically and properly processed even in heterogeneous environments. In this situation, XML digital signatures and encryption standards are meaningful as they are the only way to obtain trustworthy XML-based communications. In this paper, we describe the standard proposals coming from the W3C and the IETF, and two basic e-commerce frameworks where these standards have been applied. The first one is ebXML, which is in the definition phase and is being created by OASIS and UN/CEFACT; the second one is x-SPEED, a payment protocol that has been defined and implemented at the University of Murcia by our own research group.

Keywords: XML, Digital Signature, Ciphering, W3C, IETF, ebXML, SOAP, x-SPEED.

1 Introduction

Nowadays, XML is becoming an effective mechanism for exchanging data through the Internet, and e-commerce is one of the areas in which XML is playing a decisive role. An important aspect to take into account in an e-commerce scenario is the provision of communications security in order to prevent unauthorized parties getting or modifying sensitive or confidential information. How can we add security to XML? A simple way to cope with this is by considering an XML document as an ordinary binary document, and applying to it any cryptographic mechanism for signing or encrypting documents. But this method is not flexible, because it only enables signing or encrypting complete XML documents. What happens, for example, if we want to encrypt different fragments of a document which are to be sent to different recipients, or signing portions of a document by different senders, or applying signing and encryption simultaneously over parts of a document in any order?

We can find the answer to all these issues in the XML Signature and the XML Encryption emerging standards being developed jointly by the IETF and the W3C. These standards describe how to apply digital signatures and encryption over XML in a really flexible and powerful way.

2 The XML Digital Signature

The XML Digital Signature [XML-DSIG] can be applied to any data structure, including binary documents, complete XML files, and data fragments. In addition, it can be applied simultaneously to the contents of several documents.

Antonio F. Gómez-Skarmeta graduated in Computer Science from the Universidad de Granada (Spain), later receiving a doctorate in the same subject from the University of Murcia (Spain). Since 1993 he has been Full Professor in the department of Information and Communications Engineering of the Universidad de Murcia. He has worked on a number of research projects both in the area of distributed artificial intelligence (project M2D2) and in the area of e-commerce (PISCIS), as well as in tele-teaching and collaborative work (PUPITRE-NET) and new advanced broadband services (SABA). He is currently taking part in several European projects in the field of tele-teaching and distance learning COLAB and ITCOLE, as well as the implementation project for IPv6 EURO6IX in which he coordinates security aspects. He has published more than 40 papers internationally. <skarmeta@dif.um.es>

Maria-Encarnación Martínez-González is a Computer Science Engineer from the Universidad de Murcia (Spain). She is currently doing research and developing in the same University. Her main areas of interest are XML and security in communications. <encarna@dif.um.es>

Eduardo Martínez-Graciá is a Computer Science Engineer from the Universidad de Murcia (Spain). He is currently Assistant Professor in the Department of Information and Communications Engineering of this University. His research work is centred on the field of distributed applications applied to tele-learning, including XML, CORBA, and QoS in multimedia data transmission applications. <edumart@um.es>

Gregorio Martínez-Pérez graduated as a Computer Science Engineer from the Universidad de Murcia (Spain). In 1997 he began working in the IT Service of the same University on several projects related to security in communications. In 1999 he started working as a researcher in the Department of Information and Communications Engineering of the Universidad de Murcia, later becoming a teacher in the same department in the year 2001. His research activity is centred on security infrastructure, intelligent cards, access control systems and the new version of the IPv6 protocol, all subjects on which he has worked and continues to work via different national and international projects. He has also been published, at a national and international level, in conferences and journals. <gregorio@dif.um.es>

```
<Signature ID?>
<SignedInfo>
<CanonicalizationMethod/>
<SignatureMethod/>
  (<Reference URI? >
<Transforms>)?
        <DigestMethod>
        <DigestValue>
</Reference>)+
</SignedInfo>
<SignatureValue>
  (<KeyInfo>)?
  (Object ID?)*
</Signature>
```

Figure 1: The Generic Structure of the Signature Element

When signing XML data, there is an important issue to bear in mind: XML flexibility. By definition, XML lets us express the same information in several ways, so that two XML documents may be logically equivalent, but they may have minor differences that do not affect their logical structure, such as white spaces within tags, line delimiters, comments, and so on. XML applications tend to ignore those differences because they have no impact on the information itself, but it is critical for digital signatures because the validation of a digital signature should be performed over exactly the same stream of bytes as it was generated.

In order to solve this problem, the concept of *canonicalization* of XML documents [XML-C14N] has been introduced. In this way, converting an XML document to its canonical form consists of applying a set of transformations over it in order to represent the changeable information in a standard way. So, two documents logically equivalent will have absolutely the same canonical form.

Besides this, it is important to mention that there are several types of XML signatures. We have an *enveloping signature* when the signature contains the signed object, an *enveloped signature* if the signed object contains the signature, and a *detached signature* when the signed object is not included in the signature element.

To create an XML signature over an arbitrary digital content, a digest value of each data object that we want to sign is first obtained. Then, the resulting set of digest values is included within an XML element and joined to other information that we will detail later, and finally, a digest value of the element we have just mentioned is generated and it is signed with a private key. In this way, the validation process involves two steps: first, the signature itself is validated, ensuring the integrity of the digest value of the whole signed element, and then the digest value of each data content is validated.

XML digital signatures are represented by the *Signature* element, which has the structure shown in Figure 1. We use "?" to denote zero or one occurrence, "+" to denote one or more occurrences and "*" to denote zero or more occurrences.

As we can see, the *Signature* element contains four sub-elements: *SignedInfo*, *SignatureValue*, *KeyInfo* and *Object*. The *SignedInfo* element is the information that is actually signed. This element also includes information about the *canonicalization* method and signing algorithm used to calculate the *SignatureValue* field. *SignedInfo* contains a *Reference* element for each signed data object, which includes a reference to the data object (through an URI), the method used to calculate the digest value of this data object and the corresponding digest value. A *Reference* element may also contain some transformations that, when applied to the identified data object, produce the input to the digest operation (an example of such a transformation is the *canonicalization* method explained before).

The *KeyInfo* element indicates the key to be used to generate (and validate) the signature. It can include the keys, the key names, the certificates and any other cryptographic information of interest. Finally, the *Object* element is used to include any other data objects within the *Signature* element, which may or may not be signed. Figure 2 shows an example of an XML digital signature.

3 The XML Digital Encryption

Similar to the XML signature, the XML Encryption [XML-Enc] can be applied to any data structure, complete XML document or simple XML content element. After an XML encryption process, we obtain an *EncryptedData* element containing, through a sub-element or a reference, the data to be encrypted. If this data is an XML element, *EncryptedData* replaces it in the encrypted version of the final XML document.

```
<Signature xmlns="http://www.w3.org/2000/09/xmldsig#">
  <SignedInfo>
    <CanonicalizationMethod Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
    <SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#dsa-sha1"/>
    <Reference URI="http://www.um.es/mydocument/document.xml">
      <Transforms>
        <Transform Algorithm="http://www.w3.org/TR/2001/REC-xml-c14n-20010315"/>
      </Transforms>
      <DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1"/>
      <DigestValue>j6lwx3rvEPO0vKtMup4N</DigestValue>
    </Reference>
  </SignedInfo>
  <SignatureValue>MC0CFFrVLtRlk=...</SignatureValue>
  <KeyInfo>
    <KeyValue>
      <DSAKeyValue>
        <P>...</P><Q>...</Q><G>...</G><Y>...</Y>
      </DSAKeyValue>
    </KeyValue>
  </KeyInfo>
</Signature>
```

Figure 2: An Example of an XML Digital Signature


```
<EncryptedData Id? Type?>
  <EncryptionMethod/>?
  <ds:KeyInfo>
    <EncryptedKey>?
    <AgreementMethod>?
    <ds:KeyName>?
    <ds:RetrievaMethod>?
    <ds:*>?
  </ds:KeyInfo>?
  <CipherData>
    <CipherValue>?
    <CipherReference URI?>?
  </CipherData>
  <EncryptionProperties>?
</EncryptedData>
```

Figure 3: The Generic Structure of the EncryptedData element

Figure 3 provides a deeper view of the structure of the *EncryptedData* element.

As we can see in this figure, *EncryptedData* contains four main elements. The first of them, *EncryptionMethod*, indicates the encryption algorithm used, while the second one, *ds:KeyInfo*, identifies the key used to encrypt the data. *CipherData* contains the encrypted data or a reference to their location and, finally, the optional *EncryptionProperties* element provides other information about the encrypted data generation, such as a time-stamp.

It is also important to mention, that in some cases the *Encryption Method* and *ds:KeyInfo* elements can be optional, as the sender and receiver might have agreed in advance on the encryption algorithm and key. The *ds:KeyInfo* element offers different ways of providing the encryption key, either encrypted in an *EncryptedKey* element or through a key management system – e.g. Diffie-Hellman – by the *AgreementMethod* element; other ways are to provide the identifier of a known key in the *ds:KeyName* element, or a reference to the location where the key is stored – as in DNSsec systems – through the ds:RetrievalMethod element.

In Figure 4, we can see an example of the encryption element using a symmetric key. In this XML document, the element in plain text is replaced with the *EncryptedData* element, as we explained before. In this example we use the *Type* attribute to indicate that the encrypted data is an element. The algorithm which is applied is *3DES-CBC* and it uses a symmetric key called *My Key* to encrypt the data. The *CipherData* element contains the encrypted byte sequence using a *base64* codification within its *CipherValue* sub-element.

4 The Combination of the XML Signature and the XML Encryption Standards

The application of both standards, XML Signature and XML Encryption, over portions of an XML document may cause ambiguity about how to perform the subsequent decryption and/or verification process.

In fact, when we verify a signature, we should know if this signature was computed over the encrypted or unencrypted representation of the data. Besides this, if some encryption operations are performed over part of an already signed content, it will be necessary, before verifying, to decrypt those portions that were encrypted after signing.

To indicate the proper decryption/verification order when we use XML signature and encryption simultaneously, a new XML signature transform [XML-DSIG-Decrypt] named *decryption transform* is proposed. This element takes as an input parameter, a list of references to data portions that were encrypted before signing. This method implies that, when the receiver finds a decryption transform within a signature element, it will decrypt all the encrypted contents of the document, i.e. those that are referenced in the list included in the decrypt transform.

5 The ebXML standard

One of the most interesting initiatives of e-commerce standards employing an XML signature is ebXML [ebXML], proposed by OASIS and UN/CEFAT. ebXML uses the Simple Object Access Protocol (SOAP) [SOAP] as a messaging subsystem. SOAP is a W3C standard, which defines how message packaging should work over XML; messages are transported by protocols like HTTP or SMTP, using MIME types to identify them properly. Figure 5 shows the XML structure of a SOAP message.

In ebXML, the interaction between business information systems is based on the use of a central registry acting as a directory. Companies submit to this registry a document named Collaboration Protocol Profile (CPP), which indicates their Business Processes and Interfaces (similar to WSDL web services). Thus, the type of business between companies is dynamically determined by comparing their CPPs. This process

```
<EncryptedData xmlns='http://www.w3.org/2001/04/xmlenc#'type='http://www.w3.org/2001/04/xmlenc#Element'/>
  <EncryptionMethod Algorithm='http://www.w3.org/2001/04/xmlenc#3des-cbc'/>
  <ds:KeyInfo xmlns:ds='http://www.w3.org/2000/09/xmldsig#'>
    <ds:KeyName>My Key</ds:KeyName>
  </ds:KeyInfo>
  <CipherData>
    <CipherValue>DEADBEEF</CipherValue>
  </CipherData>
</EncryptedData>
```

Figure 4: An Example of an XML Encryption

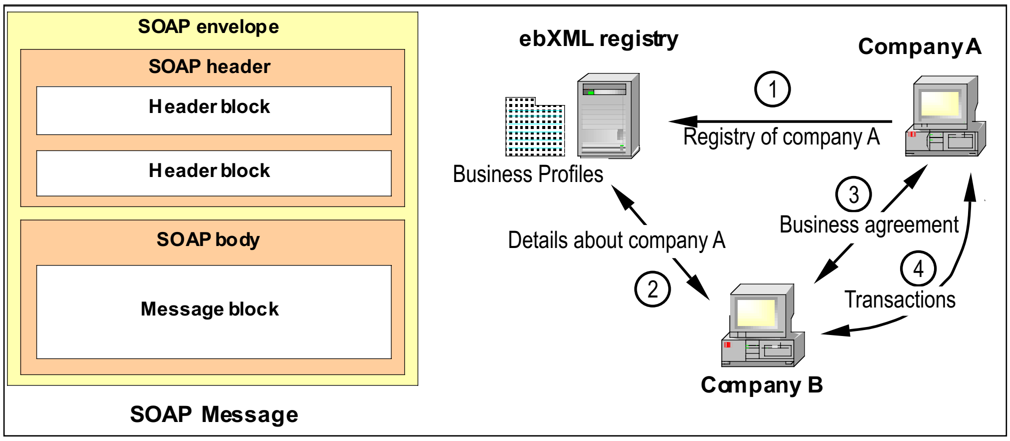


Figure 5: SOAP Message Format, and Basic Interaction with ebXML

generates a document called Collaboration Protocol Agreement (CPA), which is properly registered. The message transmission process is performed using the ebXML message service, defined over SOAP packets. In order to clarify these concepts, Figure 6 shows a SOAP packet transporting ebXML information with a signature. XML signatures appear within the SOAP message header, while signed objects are located in another part of the multipart message, referenced from the *eb:Manifest* element of the SOAP message body.

```
--MIME_boundary
Content-Type: text/xml; charset=UTF-8

<?xml version="1.0"?>
<SOAP-ENV:Envelope>
   <SOAP-ENV:Header>
      <eb:MessageHeader>
      </eb:MessageHeader>
      <Signature>
         <SignedInfo>
             <Reference URI="cid:xxx">...</Reference>
               </SignedInfo>
       </Signature>
   </SOAP-ENV:Header>
      <SOAP-ENV:Body>
         <eb:Manifest eb:id="Mani01" eb:version="2.0">
         <eb:Reference xlink:href="cid:xxx
                          xlink:role="http://ebxml.org/gci/invoice">
         </eb:Reference>
      </eb:Manifest>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>

--MIME_boundary
Content-Type: text/xml; charset=UTF-8
Content-ID: xxx

<Invoice>
   <InvoiceData>...</InvoiceData>
</Invoice>
```

Figure 6: A SOAP Message with ebXML Content Digitally Signed

6 A New XML Payment Protocol: x-SPEED

In this section we will describe a payment protocol, which is the result of research work performed by our own group. This protocol, called SPEED, i.e. *Smartcard-based Payment with Encrypted Delivery* [Ruiz-Martínez et al. 2001], provides a payment system based on a smart card-implemented e-purse and the encrypted delivery of electronic products.

In a few words, a SPEED transaction transfers electronic goods from a vendor to a client, debiting the client e-purse and incrementing the vendor account balance with the value of the acquired product. SPEED defines a set of phases including price negotiation, product delivery and payment. In addition, there are two possible operation modes: the normal mode, which enables a product price negotiation process and has been designed to provide a greater number of security features (e.g. prevention of DoS attacks and the complete authentication of the entities taking part of the transaction) and, on the other hand, the fast mode, having fewer messages than the normal mode, and conceived to be used for buying goods of smaller size or scenarios with fewer security requirements.

Its initial design and implementation were based on the use of some of the most relevant standards concerning secure communications. Some of them, like X509v3 (which was used to identify parties in a purchase scenario) and WG10 [WG10 1996] (e-purse standard) remain in the new XML version of the protocol. Others, however, have been replaced in order to obtain a more comprehensive and extensible XML specification of the protocol messages with the same level of security. This is the case with ASN.1, used to specify the messages structure, and PKCS#7 used as a basic cryptographic format for exchanging protected information.

The messages of the x-SPEED protocol, which is the adaptation of SPEED to XML, are a set of messages expressed by XML documents with a well-defined format and content. Those messages are transferred between the parties involved in every transaction. It is a B2C (Business-to-Consumer) environment in which there are three participants: client, vendor and broker (a trusted third party that performs the economic compensations and resolves disputes within the business

```
<x-SPEED>
  <messageName>
    <messageNameContent>
        ...
    </messageNameContent>
    <Signature>
        ...
    </Signature>
  </messageName>
</x-SPEED>
```

Figure 7: An x-SPEED Message before Applying Encryption

```
<EncryptedData>
  <ds:KeyInfo>
  ...
  </ds:KeyInfo>
  <CipherData>
  ...
  </CipherData>
</EncryptedData>
```

Figure 8: An x-SPEED Message after Applying Encryption

model). These parties exchange business information applying XML signatures to documents. The protocol provides integrity, authentication and non-repudiation and, in addition, the XML signature flexibility lets us sign exactly the document fragment we are interested in, in a clear and concise way. On the other hand, the application of XML encryption to x-SPEED transactions provides confidentiality and, as in the signature case, we can encrypt exactly the fragments we want.

Figure 7 shows the basic generic structure of an x-SPEED message, before applying encryption. We replace the *messageName* string with the real name of the actual message. Each message basically consists of its content, represented by the *messageNameContent* element (which will contain the protocol data to be exchanged between the parties) and a signature applied over that content, represented by the *Signature* element. This signature will be calculated using the sender private key or using a symmetric key agreed between sender and receiver.

To provide confidentiality, as a whole, to the document shown in Figure 7, x-SPEED will create a new document shown in Figure 8. In this new document, the root element will be the *EncryptedData* element from the XML Encryption specification, which we have already explained in the encryption section of this article.

The basic structure of an x-SPEED transaction that we have just shown is a simplification of the real type of message of this protocol. In fact, in most cases it is necessary to encrypt selectively different fragments to different receivers, or to have in the same document fragments signed by different senders, combining signing and encrypting simultaneously over the same data. XML signature and XML encryption specifications permit the expression of all these possibilities and combinations in a simple, clear, extensible and unambiguous way, so that any application that receives an x-SPEED message will know exactly what parts are signed, what parts are encrypted and which is the decryption/verification order.

References

[Cánovas-Reverte et al. 2001]
Óscar Cánovas Reverte, Antonio F. Gómez Skarmeta, and Gregorio Martínez Pérez, "PISCIS: Comercio Electrónico Basado en Infraestructuras de Certificación Avanzadas y Sistemas de Tarjeta Inteligente", Ist Spanish Conference on E-Commerce (SNE'01), Málaga, Spain, October 2001

[ebXML]
ebXML web site
<http://www.ebxml.org>

[Housley et al. 1999]
R. Housley, W. Ford, and D. Solo, "Internet Public Key Infrastructure, Part I: X.509 Certificate and CRL Profile", Request for Comments (RFC) 2459, IETF, January 1999

[Ruiz-Martínez et al. 2001]
Antonio Ruíz Martínez, Óscar Cánovas Reverte, Gregorio Martínez Pérez, and Antonio F. Gómez Skarmeta. "SPEED protocol: Smartcard-Based Payment with Encrypted Electronic Delivery", Information Security Conference 2001 – ISC'01, Málaga, Spain, October 2001

[SOAP]
Simple Object Access Protocol web site
<http://www.w3.org/TR/SOAP>

[WG10 1996]
CEN/TC224/WG10, "Inter-sector Electronic Purse, Part 2: Security Architecture", prEN 1546-2, January 1996

[XML-C14N]
Canonical XML
<http://www.w3.org/TR/2001/REC-xml-c14n-20010315>

[XML-DSIG]
XML Signature Syntax and Processing
<http://www.w3.org/TR/2002/REC-xmldsig-core-20020212/>

[XML-DSIG-Decrypt]
Decryption Transform for XML Signature
<http://www.w3.org/TR/2002/CR-xmlenc-decrypt-20020304>

[XML-Enc]
XML Encryption Syntax and Processing
<http://www.w3.org/TR/2002/CR-xmlenc-core-20020304/>

Realities and Possibilities of XML in the Standardisation of Digital TV with MHP (Multimedia Home Platform)

Alberto Gil-Solla, José J. Pazos-Arias, Cándido López-García, Manuel Ramos-Cabrer, José-Carlos López-Ardao, and Raúl F. Rodríguez-Rubio.

The versatility of the XML language is one of its main strengths, and because of this it is well suited to being used in a very wide range of applications that entail structuring, exchanging and processing information. This versatility has favoured its widespread acceptance, a key advantage for its introduction into the standardization processes. In this paper, this versatility is described in three activity fields related to the standardization of interactive digital TV: in the specification of the MHP standard, in the specification of the metainformation describing contents and in the scope of the implementation of applications.

Keywords: XML, Interactive Digital TV, DVB, MHP, TV-Anytime.

1 Introduction

In coming years, the use of digital technology will promote a significant change in the television world, where the viewer will move from a passive role to using the interactive facilities increasingly offered by providers. Digital technology permits us to mix traditional audiovisual contents with data, enabling the transmission of multimedia software applications to be executed in a digital television or in an analog television equipped with a decoder or Set-Top Box (STB). These applications can be synchronized with traditional content and provide interactivity to the user, together with a return channel for communication with the provider.

Nowadays, the main challenge that arises in the transition from analog to digital television is not the resolution of technological problems, but achieving the economic viability of some projects whose magnitude and scope imply huge investments both in development and technology acquisition (hardware and software) and in contents generation. Reducing the risk of these investments means reaching the number of users that permits

Alberto Gil-Solla received his degree in Telecommunication Engineering in 1991 and his Ph.D. in Telecommunication in 2000, both of them from the Universidad de Vigo (Spain). Currently, he is Associate Professor in the Telematic Engineering Department of this University. His research interests focus on the languages and protocols of Internet and the WWW, as well as on the standardization of interactive digital TV. <agil@det.uvigo.es>

José J. Pazos-Arias received his degree in Telecommunication Engineering in 1985 and his Ph.D. in Telecommunication in 1995, both of them from the Universidad Politécnica de Madrid (Spain). He has been Associate Professor in the Telematic Engineering Department of the Universidad de Vigo (Spain) since 1988, where he has been working on the application of formal methods in the Software Engineering field. Currently, his research interests are also focused on the languages and protocols of Internet and the WWW, as well as on the standardization of interactive digital TV. <jose@det.uvigo.es>

Cándido López-García received his degree in Telecommunication Engineering in 1988 and his Ph.D. in Telecommunication in 1995, both of them from the Universidad Politécnica de Madrid (Spain). He has been Associate Professor in the Telematic Engineering Department of the Universidad de Vigo (Spain) since 1988, where he has been working on performance evaluation of communication networks and on the standardization of interactive digital TV. <candido@det.uvigo.es>

Manuel Ramos-Cabrer received his degree in Telecommunication Engineering in 1991 from the Universidad Politécnica de Madrid (Spain) and his Ph.D. in Telecommunication in 2000 from the Universidad de Vigo (Spain). He has been Associate Professor in the Telematic Engineering Department of the Universidad de Vigo since 1991. His research interests are focused on the development of telematic services using conventional communication networks and digital television networks. <mramos@det.uvigo.es>

José C. López-Ardao received his degree in Telecommunication Engineering in 1990 and his Ph.D. in Telecommunication in 1999, both of them from the Universidad de Vigo (Spain). He has been Associate Professor in the Telematic Engineering Department of this University since 1990. His research interests focus on performance evaluation of communication networks and on the standardization of interactive digital TV. <jardao@det.uvigo.es>

Raúl F. Rodríguez-Rubio received his degree in Telecommunication Engineering in 1991 and his Ph.D. in Telecommunication in 2000, both of them from the Universidad de Vigo (Spain). He has been Associate Professor in the Telematic Engineering Department of this University since 1994. Previously, he was Associate Professor in the Universidad de Valladolid (Spain). His research interests are focused on performance evaluation of transport protocols and on the standardization of interactive digital TV. <rrubio@det.uvigo.es>

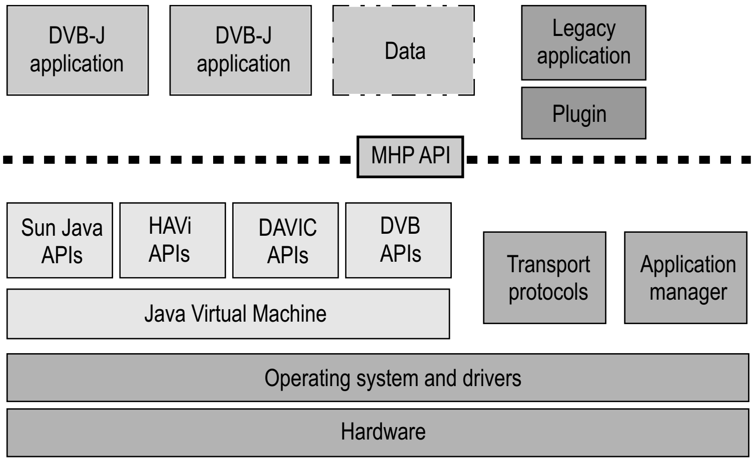


Figure 1: Architecture of a MHP Receiver

financial viability to be achieved in a reasonably short period. It implies optimizing the relationship between the added value offered to users and the price they must pay for it. And to achieve this aim, it seems necessary to standardize signals and equipment:

- As regards the transmission format of information, the technology existing today is uniform and based on the MPEG-2 standard [MPEG];
- However, for conditional access systems, interactive television services and Internet access, the solutions are proprietary, involving higher costs and equipment incompatibilities.

In Europe, the most important standards organisation is the DVB (Digital Video Broadcasting) consortium. This organisation, constituted in 1993 by the main companies involved in digital TV, arose with the objective of standardizing all technical aspects and thus sharing development efforts and reducing costs. In 1996, the European Union promoted the work of DVB in the field of standardization of the user terminal, and at the beginning of 2000 the first result was published: the MHP (Multimedia Home Platform) [MHP 1.1. 2001] specification that offers a solution for running interactive applications and offering Internet content at the television receiver.

The authors of this paper have been working during recent years on a line of research devoted to studying the convergence between the personal computer and television. One of the main objectives is the development of the software for a MHP terminal and the evaluation of its suitability to efficiently solve the outlined problems.

The rest of the paper is organized as follows: Section 2 reviews the main characteristics of the MHP standard, Section 3 describes the role of XML and, finally, Section 4 summarizes the conclusions.

2 The MHP standard

MHP specifies a generic common interface for the applications that will be executed in the decoder, providing an abstract model for access to information streams, events, files, data records and hardware resources. The API provided to the applications by MHP is independent of the hardware of the receivers. In a basic system we can distinguish several layers for processing this information (Figure 1): hardware resources and software resources dependent on the hardware; system software independent of the hardware; and applications.

From the point of view of the applications, this interface lies between them and the system software, and it consists of an execution interface and a set of APIs. These APIs facilitate the development of portable applications that can execute their life cycle and receive, display and send information regardless of the hardware platform, with only the requirement of the MHP interface.

In order to offer an abstract execution environment to the applications attached to the audiovisual information, the STB is turned into a virtual machine from the perspective of the applications. That is to say, the applications will not be developed in the native code of the microprocessor, but rather they will be Java classes independent of the hardware. So, a MHP application (known as DVB-J application or Xlet) is the bytecode of a program, written in the Java language, which fulfils two main conditions:

- It uses only the libraries and Java APIs defined in the MHP standard (generally adapted and derived from existing interfaces like Personal Java, Java Media Framework, Java TV, HAVi or DAVIC).
- It generates and handles a set of signals implementing a life cycle precisely specified in the MHP standard, and thus enables the application manager of the MHP machine to control the applications easily.

2.1 Convergence of interests and objectives between MHP and XML

The main goal of MHP is the standardization of the execution interface to be found by interactive applications when arriving at the decoder. The search for compatibility, for independence of software from the hardware platform, and for the abstraction of the format used to store data, are objectives that frequently arose in the recent history of computer applications. These objectives play a prime role in the MHP standard and are the cornerstone of the strategy to achieve a new technological framework in a market in great need of stability and characterized by great inertia, but also by a constant obligation to provide compatibility.

In this context, XML is an extremely suitable tool to help achieve the proposed goals due to its advantages regarding formality, extensibility, universality, structuring capacity, platform independence, widespread acceptance, great availability of support tools and simplicity in carrying out the processes of information verification. It is not just that XML and MHP share fundamental aims; or that the aforementioned technical advantages of XML are appropriate to solve the problems that emerge around MHP; but also the fast expansion and widespread acceptance of XML places this technology in a privileged position to serve as the dissemination vehicle for any standardization attempt, and especially for MHP.

3 Use of XML in the environment of MHP

XML can be used in three activity fields in the standardization of digital television with MHP, by playing an important role: in the specification of the MHP standard, in the specification of the metainformation describing contents and in the scope of the implementation of applications.

3.1 In the MHP standard

First of all, we must emphasize the extensive use of XML in the expression of the mechanisms defined in the MHP standard itself, what could be described as the infrastructure of the system. There are many situations in the specification regarding the exchange of information among entities that are solved via the definition of XML applications that structure the data to be sent. These applications are generally small sized, and designed to resolve particular situations by reusing software belonging to other tasks. In all cases, the formalization of the defined languages is carried out via DTDs and not via Schemas [W3C 2001], given that these applications had already been specified in the first version of the standard (MHP 1.0) that dates from the beginning of 2000, prior to the specification of the W3C XML Schema 1.0.

Next, let us review some of the aforementioned situations.

3.1.1 Specification of permissions to access resources

In the function of the digital TV, the STB plays an active role, mainly characterized by its capacity to execute software received from the network. Obviously, this generates a set of problems arising from the need to authenticate the downloaded software and from the implementation of suitable mechanisms to guarantee that those applications get access to decoder resources.

In the MHP standard, the negotiated permissions given to an application are the result of the intersection between the permissions requested by the operator sending the application and the permissions specified by the user of the STB. This latter mechanism is free and dependent on the implementation. However, the first one is explicitly defined in the standard and it is implemented via XML files where all the parameters of the request are specified.

The standard defines the DTD of an application called *XML Permission Request File* and it specifies that a document of this type can be sent with each application for describing the requested permissions. This application includes the elements and necessary attributes to request access to decoder resources such as the file system, the life cycle of other applications, the return channel, other television channels, the user configuration, the conditional-access mechanisms, the connections with remote computers, etc.

3.1.2 Applications amenable to being locally stored

One of the most interesting characteristics of digital television is the possibility of the local storage of programs and/or applications, in such a way that the user can watch programs (and execute their associated applications) in a non-linear way.

With respect to the associated applications, the possibility of storing a DVB-J application is indicated in the AIT (*Application Information Table*), a data structure that defines important parameters regarding the capabilities of the applications. In this case, the operator provides, together with the application classes, a XML file which lists all the components to store. The DTD of this application, known as *Application Descriptor File*, indicates the classes and directory to save, as well as additional information like size and priority.

3.1.3 DVB-HTML: an application of XML

The high cost of the development and installation of technologies needed to transform television into an active platform demands the development of services. One of these services is undoubtedly Internet access via television. For this reason, it was included by the MHP standard as a high-priority objective [GRIS 2002]. Furthermore, the DVB always emphasized the necessity to provide a broadcast channel for structured and stylized information, like a “super-teletext” that provides a versatile channel to display information to the DVB-J applications in a homogeneous and stable way.

The subsequent work led to the adoption of XML as the backbone of the proposed solution, and to the definition of an application of XML, the DVB-HTML language, for implementing “super-teletext” and standardizing WWW pages for a MHP receiver. The result was the specification of the DVB-HTML applications.

A DVB-HTML application is a set of interrelated documents, written in DVB-HTML language, limited by an entry point and a boundary. The objective is the definition of a framework for broadcasting sets of related pages, making the access to information faster.

The DVB-HTML language is an application of XML defined by means of the selection of several modules of the *XML Modularization* standard [W3C 2001] that bring together the distinctive characteristics of the television compared with access through a computer: lower processing power and screen resolution, longer viewing distance or worse ergonomics in the user interaction. To take account of these differences, DVB-HTML includes all the modules defined in *XML Modularization*, except *complete tables*, *edit*, *legacy*, *server-side image maps*, *name identification and applet*. Additionally, it replaces the module of intrinsic events with a more specific dedicated version.

3.1.4 The declaration and association of triggers

The model based on broadcasting information via DVB-HTML applications does not try to be limited to a static outline where the user downloads and displays information structured in pages. The MHP standard enables the operator to interact with downloaded applications by sending synchronization messages to modify their life cycles. These messages, termed triggers, are sent via the broadcast channel and delivered to the DVB-HTML applications as DOM events [W3C 2001], which usually entail executing JavaScript code contained in the documents.

This mechanism has been implemented by the definition of two XML applications. In the information contained in these files, each application declares the triggers it wants to subscribe to. In the first file, called *Event Factory File*, the creator of the application associates triggers to concrete elements in DVB-HTML documents, so that they are delivered through the propagation of DOM events generated deliberately. The second one, known as *Event Linkage File*, is used to declare all the *Event Factory Files* related to a concrete DVB-HTML application and to associate them with their respective documents.

3.2 In the specification of metainformation describing contents

A second working field where XML is achieving excellent results is that of contents characterization. This use of XML was quickly identified in the scope of the WWW and it has been extensively exploited to improve (or, rather, to create) services for searching and recovering information on the Internet (mainly web pages) that satisfies user requirements.

However, the audiovisual (and multimedia in general) content of the WWW has continued to grow spectacularly in volume, and this type of information presents greater difficulties in the establishment of search criteria based on content. More and more, users demand search mechanisms and efficient recovery of this type of information starting from objective and easily identifiable characteristics. Nowadays, the only mechanism that is drawing together significant effort and achieving concrete results is the association of metainformation with these contents through XML applications that include the most representative parameters of the objects to be characterized.

In this context, it is important to highlight the role played by the MPEG-7 standard [MPEG] in the process of identifying the multimedia objects of our interest and the properties associated with them: still images, graphics, 3D models, audio streams, voice, video, etc. In the context of this standard, the *Description Definition Language* (DDL) is defined. The DDL is, basically, the *XML Schema 1.0* standard extended with several elements that facilitate the characterization of multimedia objects and situations. So, the DDL is used in the specification of descriptive outlines that characterize the various identified objects. These characterizations permit implementation of search services from parameters such as a sequence of musical notes, some characteristics of the voice, geometric objects, colours, movements, etc.

Focusing on the specific field of digital television, the most important project in contents characterization was promoted by the TV-Anytime forum [TV-Anytime]. The problem of the enormous increase in contents and the difficulty of finding whatever is sought (something quite well-known in the WWW) has motivated this alliance of companies to implement mechanisms to describe contents and to refer to them regardless of their location and broadcast date. The main goals are twofold: to separate the information that describes the contents from the information needed to locate them, and to make the whole independent of the transport mechanism.

The tool used to achieve these goals is also XML. Their implementation starts from the DDL defined by MPEG-7 to create a set of XML applications that describe all the important parameters to enable the search and selection of the contents that could be more friendly to a user. This metainformation covers traditional objectives (descriptive attributes of the contents, technical characteristics of their representation formats, revisions and criticism by a third party or information about different copies of the same content). In addition, it includes ambitious ideas about parameters that would permit the implementation of novel services, among them:

- The identification of user's characteristics: It would enable the search for more attractive contents and the carrying out of market research to guide the future generation of contents.
- The specification of characteristics of social groups that could be more or less receptive to certain types of contents: It would permit making a search in the opposite direction (contents could look for their audience and be offered to it).
- The characterization of different parts of the contents: It would enable the segmentation and indexation of the contents, thus easing the way toward a non-linear model of watching television where users can go straight to what they want.

This application field of XML shows us one of the most prominent characteristics of this technology: its extensibility. XML enables the building of standards on others in a natural way: XML *Schema* is built on XML, the DDL of MPEG-7 is built on XML *Schema*, TV-Anytime is built on DDL, and so forth.

3.3 In the scope of the implementation of applications

Implementing tools for generating, analysing, processing and displaying information is another important field in which XML can be successfully applied. In this scene, the experience of the authors centres on using XML in the specification of MPEG-2 transport streams according to DVB-MHP.

In the project described in [GRIS], with the goal of software verification, we needed transport streams that offered a representative range of the different structures of information defined in DVB-MHP. These streams were not available on the market or in research circles until the project was already advanced (in late 2000). So, we decided to implement a simulator that generated these different situations for validation purposes. XML was used in this work to structure and store the configuration information so that the application can generate streams with different characteristics.

The information structure modelled via the defined XML applications ranges from the elements that describe the components of the broadcast network to the elements of the available interactive applications in each television channel. In Figure 2 we can see the hierarchical structure of the entities to be modeled.

If we just describe the most important entities, we can assign the role of root element in our system to the entity "broadcast network". This element would be compounded of several transport streams that in turn would be made up of the different

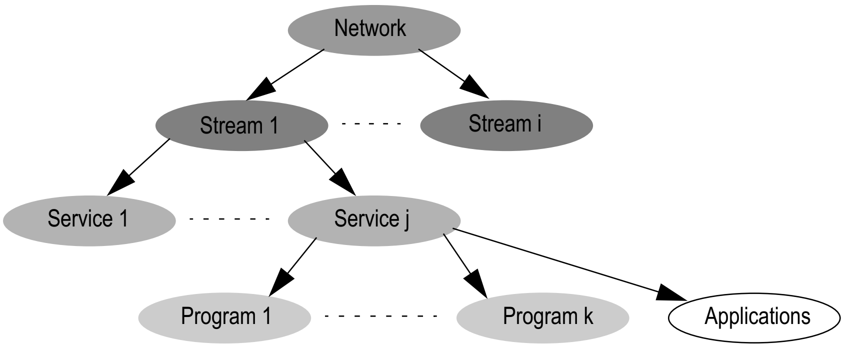


Figure 2: Hierarchical Structure of the XML Application

services (television channels) in the same stream. In each stream there are two important entities that require special signalling information: the programs of the service (sequence of audiovisual events to be broadcast) and the applications with additional functionality such as interactivity. The specified information is not only made up of the components describing the entities, but also of their dynamic behaviour when they are used (for example, configuring an application into a service as autostart or not).

The result is a mean-sized XML application that uses 32 elements and 49 attributes to specify those characteristics of the system entities to be configured. The complete characterization of the entities is a much harder task, but our XML application only needs to declare those elements that can be configured in the stream generator.

When using XML to solve the described task, we identified several advantages that we can classify into three categories:

- *Advantages inherent to XML:* In this section we have found advantages widely described in the literature: ease of organizing adequately a lot of complex information; legibility of the results, which permits use of the specifications as documents for discussion and dissemination of knowledge; extensibility that enables preliminary work with partial descriptions to be completed when necessary; etc.
- *Advantages derived from the availability of public-domain software:* In this case, our work was made much easier due to the possibility of using public-domain tools for creating and editing the large quantity of XML documents to describe the different scenarios to be verified. But even more important is the possibility of using public-domain software to read, analyse and validate the XML documents, as well as to automatically generate data structures. Here, the advantages, derived from the choice of XML and from the use of Xerces [Apache], are quite clear in the reduction of development time, software reliability and the possibility of reusing the same functionality in different tasks of the project.
- *Advantages derived from our implementation:* Actually, the developed XML application was implemented through the combination of five autonomous XML applications describing the aforementioned entities, and referring to each other according to the structure of Figure 2. Splitting the configuration of the transport stream into independent applications contributes the additional advantage of reusing specifications in other tasks of the project. Specifically, the XML application that describes the program scheduling broadcast into the transport stream on different days is used to make an EPG (Electronic Program Guide) that offers the appropriate query services in the STB. As another example, the XML application characterizing the DVB-J applications that implement the interactive facilities is reused in the tools for the automatic generation of these DVB-J applications, on which our research group is now working.

4 Conclusions

Television is undoubtedly one of the sectors where standards enjoy greater stability, since the fundamental characteristics of equipment and signals change little over time, and the changes always observe the standards for compatibility. The technological transformation from the analog to the digital world must observe this principle, integrating all of the existing technologies with the new concepts.

In this context, XML is an extremely suitable tool to help achieve the proposed goals due to its advantages in formality, extensibility, universality, structuring capacity, platform independence, widespread acceptance, great availability of support tools and simplicity to carry out processes of information verification. Furthermore, the technological model chosen for the implementation of interactive digital television (by both DVB and ATSC) is intimately related to Java and nobody is surprised at the close connection between the two technologies, Java and XML, from the beginning. Any standard or recommendation associated with XML has numerous support tools developed in Java at its disposal, as well as a wealth of free Java software.

The versatility of the XML language is one of its main strengths. As we have described in this paper, it is a technology with a wide range of applications, from the standardization of the infrastructure to the characterization of information sent through that infrastructure.

With regard to the characterization of information, the use of XML is still in its early stages. The great number of channels in digital TV implies a considerable increase in information available to the subscriber. Undoubtedly, this will mean that the problem of searching and selecting content will become much more complex in years to come. In this scenario, content characterization through metainformation seems an inevitable way to implement automatic mechanisms that help the user to identify content.

Although TV-Anytime does not result in a definitive standard on this matter, the ease provided by XML to define automatic transformations among the current formats, and those that can appear in future, guarantees XML a privileged position in this scenario.

References

[Apache]
The Apache Software Foundation, Xerces, <http://www.apache.org>

[GRIS]
Grupo de Redes e Ingeniería del Software (GRIS). Depto. de Ingeniería Telemática. Universidad de Vigo (Spain). Sistema de recepción digital multimedia, <http://mhp.det.uvigo.es>

[GRIS 2002]
GRIS Research Group. Internet-TV convergence in DVB-MHP. In VIPromCom, Zadar (Croatia) 2002

[MHP 1.1. 2001]
DVB Consortium, Multimedia Home Platform 1.1, http://www.mhp.org, July, 2001.

[MPEG]
Motion Picture Experts Group, <http://mpeg.cselt.it>

[TV-Anytime]
The TV-Anytime Forum, <http://www.tv-anytime.org>

[W3C 2001]
W3C Consortium, <http://www.w3.org>, April, 2001.

Application of XML to the Journalism Field

Luis Sánchez-Fernández, Carlos Delgado-Kloos, Vicente Luque-Centeno, María del Carmen Fernández-Panadero, and Laura Martínez-Bermejo

Journalism is a typical example of an application domain in which the use of XML is changing the work processes. Currently news items are stored and exchanged using proprietary formats. These formats are becoming obsolete, due in part to the development of Internet and multimedia technologies. In these circumstances the International Press Telecommunications Council (IPTC), that is composed of the organizations and companies concerned with news collection, distribution and publishing around the world is standardizing new formats based on XML to improve current work processes in this field. In this article we present the work that we, and representatives from the Department of Information Science of our university, have carried out in this field. Our work is based on a model we have developed for the work processes in the journalistic field based on XML.

Keywords: XML, Journalism, Multiplatform Support, Personalization, E-Commerce.

1 Introduction

In the 1970's advances in computing and telecommunications to exchange and to store documents led to computerization of the work processes many fields, including journalism.

At that time specific news description formats were developed, e.g. ANPA 1312 [Wire 1979] and IPTC 7901 [IPTC 1979], to enable documents to be exchanged and stored electronically. The formats were processed by applications developed specifically for that purpose.

A news item encoded in one of these formats contains the text of the news item (only textual news are supported), together with additional index category fields e.g. identification number, date, section, priority.

The evolution of technology (Internet, multimedia etc.), and increased journalistic activity, is making these formats progressively obsolete. Identified deficiencies of these formats include:

- They do not support the description of multimedia content.
- They do not have metadata to ease the classification and retrieval of news (search by people, place etc.).
- The available index categories lack detail.
- They do not support the whole life cycle of a news item e.g. it is not possible to identify the originator of a news item, to include what version of it is currently available, and who has received it.
- They do not allow links to other related news items.
- They do not allow the composition of news items (e.g. to associate a photograph with the text of a news item).

In response to these needs the journalistic standards organizations decided to develop new standards based on XML applications.

In this article we present a process model that we propose for future journalistic activity. This model has been developed

Luis Sánchez-Fernández is a Full Professor in the Department of Telematics Engineering of the Universidad Carlos III of Madrid (Spain). He is a Doctor of Telecommunications Engineering from the Universidad Politécnica de Madrid. He is the author of more than 30 publications in journals and national and international conferences. He has taken part in several national and international research projects. Currently, his main interest is in the following fields: electronic publishing, electronic journalism and XML. He is member of ATI (Asociación de Técnicos de Informática, Spain.) <luis@it.uc3m.es>

Carlos Delgado-Kloos is presently a Full Professor at the Universidad Carlos III of Madrid (Spain), where he also holds the posts of Director of the Department of Telematics Engineering and Director of the Master in E-commerce. His current interests include design languages and techniques, as well as Internet-based applications such as electronic publishing, tele-education and e-commerce. He has been involved in many projects with European, national (Spanish Ministry) and bilateral (Spanish-German and Spanish-French) funding. He has published over 60 articles in national and international conferences and journals, and has also written one book and co-edited four more. He holds or has held various posts in national and international bodies such as: vice-president of IFIP TC 10, secretary of IFIP WG 10.5, editor of the Springer journal 'Formal Aspects of Computing', assistant director of Telecommunication Engineering at his University and manager of the National Programme for Information and Communication Technologies at the Spanish Ministry for Science and Research. He has been programme committee member or chair at more than 40 conferences and workshops, including vice-programme chair for the IFIP'92 World Computer Congress and programme chair for DATE 2002. He belongs to several professional associations in Spain and abroad, ATI being one of them, and has published several papers in Novática, ATI's journal whose Editorial Board he was a member of.
<cdk@it.uc3m.es>

(See the CVs of the other authors in the following page.)

during two research projects “El Periotrónico” and “Infomedia”. In section 2 we present XML applications for journalism which have been developed by the IPTC [IPTC]. In sections 3 and 4 the model and its implementation are described. Finally section 5 presents the conclusions and identifies areas for further work.

2 NITF and NewsML

Being aware of the limitations of the current news description formats the IPTC has developed new formats based on XML [XML 2000].

NITF [NIFT] was the first XML standard developed by the IPTC in the 1990’s. NITF is a XML application for the description of textual news items, although it may contain multimedia objects. Incorporating many metadata elements this language addresses some of the deficiencies that had been identified in the previous standards.

The requirement to support multimedia technologies has resulted in a new standard, NewsML [NewsML 2000], being developed. NewsML attempts to address all the deficiencies of current news formats identified in the introduction. NewsML makes use of complementary languages e.g. NITF can be used for the description of textual news inside a NewsML description. The main characteristics of NewsML are:

- It introduces the concept of a Topicset which allows the definition of a set of keywords that can be applied to describe sections of news items e.g. cities, people etc. Being a flexible format NewsML is capable of further development.
- It supports all types of media and digital formats. To achieve this the format and the media are defined by means of Topicsets; these are capable of being extended to support any new formats developed in the future. The media content itself can be included in the NewsML document or linked with a Uniform Resource Locator (URL).
- It allows the addition of a metadata to a multimedia news item e.g. other details of people who feature in the news item, or details of the location the news item refers to. By not fixing the format of the content it allows the use of auxiliary languages to describe the content e.g. NITF is recommended for textual content.
- It allows version control of a news item.
- It allows the identification of the originator, dealer and receiver of a news item.
- It allows the relationships between news items to be defined.
- It allows collections of news items to be defined.
- It allows the definition of news items that are themselves composed of other news items (e.g. text and image).
- It supports alternative reports of the same news item (e.g. news written in different languages).

NITF and NewsML are periodically updated. Currently version 3.0 of NITF and version 1.0.1 of NewsML are available.

3 Model of Journalistic Activity Based on XML

Journalistic activity can be divided in three tasks:

1. Creation of a news item by a reporter (in many cases on behalf of a news agency).
2. Selection of news items to publish (usually made by a media organization).
3. Binding the news items together to create a newspaper or an electronic newspaper.

Currently the news content produced in the news agencies is mainly in textual format, usually made available to the media by means of databases. The news items are created using a news description format (e.g. ANPA 1312) that, among other limitations, does not support multimedia content.

Having selected which news items to publish from the news items made available by agencies and other media sources a media organization decides the layout of the newspaper, either paper or electronic version.

The model that we propose is presented in Figure 1. In this model the news agency will become a multimedia information agency able to produce news content in all digital formats (text, image, audio, video, computer graphics, computer animation). The journalistic content will be marked using the XML standard application and archived in a XML database which will contain current and past news content.

The News media will be able to make simultaneous searches over the databases of different news agencies, taking advantage of the fact that they will all be using the same news format. The metadata support offered by NewsML will allow more intelligent and efficient searches.

The newspaper will be created from news content in NewsML format (from news agencies or in-house). By using

Vicente Luque-Centeno graduated as a Computer Science Engineer from the Universidad Politécnica de Madrid (Spain) in 1996. Since then he has been studying for his PhD. in the Department of Telematic Engineering of the Universidad Carlos III of Madrid, participating in several research projects related to electronic publishing, and authoring and publishing papers on that topic. He is an Associate teacher on the Telecommunication Engineering course. <vlc@it.uc3m.es>

María del Carmen Fernández-Panadero graduated in Physics from the Universidad Complutense de Madrid (Spain) then worked for Andersen Consulting on projects related to multimedia technology and Internet programming. Since 1999 she has worked in the Department of Telematic Engineering at the Universidad Carlos III of Madrid. Her interests include Internet-based applications, such as tele-education, and electronic publishing. Her activities at the University include undergraduate teaching, professional training for companies, and participation in several projects related to tele-education and electronic-publishing. She has been on the organising committee of several conferences (Pubelec, Jitel) and has had papers related to tele-education, electronic publishing and the application of XML standards to these fields published in national and international journals. <mcfp@it.uc3m.es>

Laura Martínez-Bermejo is studying for her MSc. in the Department of Telematic Engineering at the Universidad Carlos III of Madrid (Spain). Her work concentrates on the implementation of a software module to enable dynamic personalisation in the prototype electronic journal that is being developed under the framework of the Spanish national project “Infomedia”. <lmartinez@it.uc3m.es>

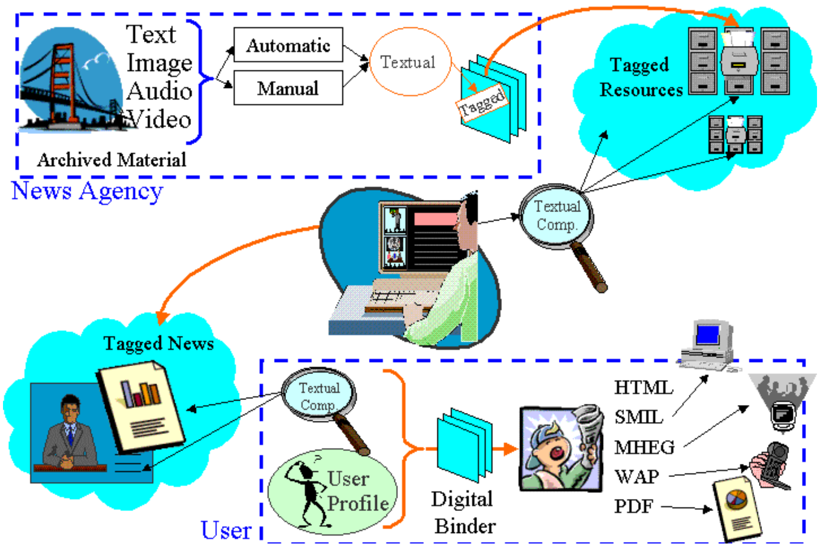


Figure 1: Model of Journalistic Activity in the Future

different style sheets it will be possible to automatically create different versions of the newspaper for different platforms: paper for traditional distributions, HTML [HTML] for Web based access, WML [WML 2002] for mobile phones and digital television.

As well as enabling a newspaper to be published in different formats NewsML could enable further personalization. The newspaper could offer the final user a daily news database that could have wider coverage than the standard newspaper. The final user could register preferences with the newspaper server and by accessing the news database the server could choose the news items that most closely match the preferences of the final user.

4 Implementation of the Model

The model that is presented in the previous section has been partially implemented through two state funded research projects: "El Periotrónico" and "Infomedia". We have worked in the implementation of this model before the appearance of NewsML, in "El Periotrónico". After NewsML became an standard, we adopted NewsML as our news description format in the "Infomedia" project, instead of the XML application Journalism Modelling Language (JML) developed in "El Periotrónico", as explained below.

4.1 "El Periotrónico"

In the "El Periotrónico" project an electronic newspaper prototype was created[Fernández et al. 1999]. In this prototype the final user was asked to complete a form from a Web browser that offered the opportunity to register some preferences for content (newspaper sections, keywords or authors) and for presentation (colours, font).

The prototype has access to a news database. Using software modules specifically developed for the project, the following tasks are carried out:

- A software module periodically dumps the most recent content of the database to files ready to be delivered by the Web server.
- A Common Gateway Interface (CGI) is used to construct the personalized newspaper. To do this, each day news items are selected from fresh news items that fit the user preferences.

In this first prototype version, news items were kept in HTML format.

Although when "El Periotrónico" project began neither NewsML nor NITF existed, we were aware of the interest of describing news in XML format. A Document Type Definition (DTD) was developed in JML allowing textual news items to be marked. In addition a simple editor was developed that allowed a journalist to create news items in JML format.

4.2 "Infomedia"

The electronic newspaper created in the "El Periotrónico" project was then used in the "Infomedia" project to implement three aspects of the model: multiplatform support, dynamic personalization and e-commerce.

Multiplatform Support

The object of this aspect of the work is to be able to create newspaper versions for different platforms from a set of news items in NewsML format. In doing so it is necessary to take into consideration that the generation of different versions changes depending on the following variables:

- *Presentation*. The main consideration is that depending on the platform being used the amount of contents that can be presented simultaneously could change. The different platforms were ordered according to the amount of contents that can be presented on them as follows: paper, computer, television, mobile. Accordingly the presentation has to be adapted to the platform characteristics. Moreover, the quality of the presentation must be taken into account e.g. leaving a single word as the last line of a paragraph must be avoided.

- *Structure*. To present this information on different platforms it is necessary to organize the information in levels. This is influenced by the previous point because if less information can be presented, more levels are needed, although for the platforms that allow less simultaneous information it is necessary to reduce the total amount of information that the newspaper contains. A structure that makes the use of the newspaper easier by assembling news for subjects and reducing the number of levels is required.
- *Content*. In addition to presentation and structure, the news contents itself can change from one platform to another, and not only in the amount of information provided. It is known that the user characteristics of different channels change, e.g. the typical Internet user is a young person who has some knowledge of computing whereas users of digital television services may be a wider collective, in most cases with a different cultural need. So contents must not only satisfy the amount of information required but also the way the information is presented.

The work that is being done consists of the design of a set of XSL [XSL 2001] style sheets that allow the transformation of NewsML into different presentation formats. XSL is one of the standards that the World Wide Web Consortium is developing around XML. A XSL document is itself a XML document that defines the transformation rules based on patterns that allow the transformation of a XML document to another format e.g. HTML, WML.

To obtain a quality presentation, it is considered necessary that a newspaper designer should guide the transformation process. A tool is being developed that will allow the definition of templates in which fields are being identified (e.g. headline, text, image etc.). These fields can be filled with contents extracted from NewsML documents. With this set of templates plus the NewsML documents, it will be possible to obtain the newspaper in the required formats.

The templates that are being constructed must allow the selection of paragraphs from a news item, selection of the required media (text, video, image, audio etc.), merging of the contents in the template and production of the final document. The templates can also include the reader preferences (personalization) and the content can be customized as NewsML is able to keep different versions of the same news item.

Dynamic Personalization

Personalization can be achieved by two different ways: static personalization and dynamic personalization. In static personalization the final user fills in and submits a form to explicitly indicate preferences for content and presentation. This type of personalization was implemented in "El Periotrónico".

Dynamic personalization is based on collecting information about the previous behaviour of the final user and building a user profile. Knowing what a final user has accessed in the past it is reasonable suppose that the final user will also be interested subjects in the future. If one day a final user reads a piece of news about a certain subject (e.g., Stock Market prices), it is likely that there will be a similar interest the next day. The implementation of this personalization type is one of the objects of "Infomedia".

Text mining techniques are used to implement the dynamic personalization module. The algorithm used is composed of 3 phases. The first phase involves text pre-processing to extract keywords from news viewed by the final user. The second phase develops a mining process by constructing a user profile – a boolean logic function – using generic algorithms. In a third phase, each day the news items that could be interesting to the final user are selected using the user profile.

To extract the keywords, different strategies can be used. We base selection of keywords on the most frequently occurring words once "empty" words (articles, pronouns etc.) have been removed, giving a special weighting to those which appear in the news title or in the lead-in.

The process of news selection is based on applying a given boolean function in which each argument of the function indicates if a keyword is present (logic value true) or not (logic value false) in the news item that is being analysed. If the logic function is evaluated to true, the news item is selected.

In order to perform an effective dynamic personalization process it is necessary to have a wide set of news items that the user has already read. We estimate that the number of news items necessary to obtain reliable results is around 100 although further experimentation is necessary. This set of news items are obtained over several consecutive days with the keywords and the boolean function being updated by adding the news items that have been most recently read by the user and removing the old ones. Dynamic personalization will be used together with the static personalization available from the first prototype.

E-commerce

It is known that the majority of electronic newspapers lose money. We consider that in the medium to long-term an electronic newspaper based exclusively on advertising revenue is not financially viable. We are developing a model where the final user is charged either for each news item accessed or for each specialised news item accessed.

5 Conclusions and Areas for Further Work

XML technology is being used successfully in the many fields in which a format is necessary for cataloguing, storing and exchanging information.

In the case of journalism the use of XML, and the tools and standards being developed around it, will in future allow journalistic information to be accessed in a most efficient and flexible way. This will offer the final user new services and new ways to access journalistic information.

We plan to continue the work, building on what has already been achieved, to address news content management with XML. This will include aspects such as support of information flow, life cycle of contents (creation, modification etc.) and efficient searches.

Acknowledgements

The work presented in this article has been partially funded by the Centro de Investigación Científica Y Tecnológica (CICYT) of Spain through the TEL97-0788 “El Periotrónico” and TEL99-0207 “Infomedia” Projects. Fruitful collaboration with Antonio Hernández-Pérez, Tomás Nogales-Flores and David Rodríguez-Mateos is gratefully acknowledged.

References

[Fernández et al. 1999]
M. Carmen Fernández Panadero, Vicente Luque Centeno, Carlos Delgado Kloos, Andrs Marín López, Carlos García Rubio, Luis Sánchez Fernández y Antonio Hernández Pérez. Mass-Customizing Electronic Journals. In Electronic Publishing'99. Redefining the Information Chain-New Ways and Voices, May 1999.

[HTML]
HyperText Markup Language (HTML) Home Page. World Wide Web Consortium. <http://www.w3.org/MarkUp/>.

[IPTC]
International Press Telecommunications Council (IPTC). <http://www.iptc.org>.

[IPTC 1979]
The IPTC Recommended Message Format IPTC TEC-7901 (IPTC 7901). International Press Telecommunications Council, 1979.

[NewsML 2000]
NewsML Version 1.0-Functional Specification. International Press Telecommunications Council, October 2000. <http://www.iptc.org/site/NewsML/specification/NewsMLv1.0.pdf>.

[NIFT]
News Industry Text Format (NITF). <http://www.nitf.org>.

[Wire 1979]
Wire Service Transmission Guidelines (ANPA 1312). American Newspaper Publishers Association, February 1979. <http://www.naa.org/technology/standard/89-3msw.pdf>

[WML 2002]
Wireless Markup Language (WML) version 2 Specification. WAP Forum. <http://www.wapforum.org/what/technical.htm>.

[XML 2000]
Extensible Markup Language (XML) 1.0 Recommendation, October 2000. <http://www.w3.org/TR/2000/REC-xml-20001006>.

[XSL 2001]
Extensible Stylesheet Language (XSL) 1.0 Recommendation, October 2001. <http://www.w3.org/TR/xsl/>.

A New Face for Each Show: Make Up Your Content by Effective Variants Engineering

Martina Hemrich

This paper presents an approach for variant engineering to address use cases found in the production of information types, like reference works and legal publishing – where the composition of variant fragments to products demands concrete, practicable information technology concepts. It will then describe two use-cases of reference works and legal publishing, the real solutions of which will be in the focus of the slides presentation at the conference. In the solutions presented we will show proven strategies for the multiple re-use of a piece of information for various products and under given aspects.

Keywords: Collection, Content Management, Effectivity, Fragments, Legal Publishing, Metadata, Modular, Multi-Channel Production, Re-purposing, Re-use, Reference Works, SGML, Smart Content, Variant, XML.

1 Introduction

Existing business cases and the ever-broadening vision of publishing markets put increasingly complex demands on the production of factual information. To achieve the business targets it is necessary to identify and satisfy these demands in a seamless editorial and content management environment. How can this be achieved considering the multiple expectations on our products that we want to meet?

One crucial feature of an effective production is certainly the minimization of redundant information. *Information re-use* is the buzzword. To achieve this, we structure our content using concepts like XML thus enabling it for *fragmentation* and *modularisation*. Applied consistently, this has a direct positive effect on production and maintenance costs.

But there's more to the production of factual content: some crucial characteristics of factual information are that it is purpose-driven, market-oriented, and user-dependent. Here, (I re-use a term I found in [Terris 2001]) *re-purpose* is the buzzword. The occurrence of a piece of information; be it text, images, animations or references, varies with the multi-faceted demands of user expectations and capabilities, publishing media, time and money. We call each *new occurrence* of an existing piece of information, for a *certain purpose* and under *certain conditions*, a *variant*.

The outcomes of an XML-based and variants aware content production will get the company another step nearer to what has been promised to be one of *the* advantages of SGML/XML – the creation of *smart content*.

On the production level:

- single-source and multi-channel publishing
- improved information quality
- enhanced functionality of electronic output

On the business level:

- new products quickly to market
- products that fit the individual's expectations
- possibility to anticipate what makes the customer happy and produce that in reasonable amount of time;

All the above leads to *increased customer loyalty.*

We define *smart content* as a:

- managed collection of product neutral, reusable, linked and well structured information objects
- plus their associated metadata,
- stored in standards-based media neutral format,
- which can be navigated, searched and combined,
- to provide timely, targeted, user adapted and traceable information to the end user in the required medium.

Martina Hemrich works as a project manager and IT consultant at empolis GmbH (formerly STEP Electronic Publishing Solutions GmbH) <http://www.empolis.com/>, her main focus lying on consulting in XML and related standards and managing the implementation of XML-based solutions realized with the empolis product family. The projects she has been involved in for the last years mostly deal with legal content and reference works. She started at STEP in 1995 specializing in DTD design and the rendering of structured data (both layout and linking concepts). <martina.hemrich@empolis.com>

2 Terms and Definitions

Since the interpretation of the keywords used in this paper often is as confusing as the matter itself, let us first introduce our understanding of them.

information object	An information object is the smallest content unit managed by a content management system.

link object A link connecting two or more information objects; link objects carry information (e.g. metadata) which can be modified without modifying any information objects being associated to them.

metadata Information about a managed object (for example validity, status, administrative issues, etc.)

managed object In this context a managed object is an object handled with versioning by the content management system. There are two managed object types: information objects and link objects. Metadata is not considered to be a managed object in this context, because metadata are seen as an integral part of a managed object and therefore is implicitly versioned.

collection of managed objects A collection of managed objects is an arbitrary set of managed objects. A collection may have a defined sequence and graph representation (e.g. container structure) of its members.

occurrence An information object or link object which contains content or other information on an abstract subject.

production A production is an editorial process instance to produce a specific version and variant of a publication.

publication An arbitrary collection of managed objects for the purpose of delivery as a product to a target group. A publication runs through a workflow and is released for delivery. Publications may be regularly published in new versions or in different variants.

re-use "using the EXACT same information object (image or text) in more than one 'document'" [Terris 2001]

re-purpose "extracting and/or formatting the same piece of information in many different ways, usually producing a different document type targeted for a different user and/or purpose" [Terris 2001]

subject everything that has been assigned a 'name'

variant A variant is a managed object or collection of such with alternative content to another managed object or a collection of such. Two associated variants may have different content and related data (such as link arcs and metadata). Variants are independently modifiable by editing. Variants are created based on decisions (human triggered). Variants know about their relationship to the other variants. Variants have some attached validity metadata, which define the context in which the variant is valid/effective (e.g. language, target medium).

variant set All the variant occurrences to a given subject/topic. A set of managed objects making up all defined variants to a managed object.

version A version is a self-contained persistent snapshot of the data and related data (such as link arcs and metadata) of a managed object or a collection of managed objects at a given time. All versions of the same origin (managed object or collection) build up a clearly defined (temporal) sequence. Versions except the current version are read only and can't be modified. The creation of versions is normally rule/event based (e.g. check-in or publication).

3 General Assumptions on Variants Engineering

Imagining the constant creation of new variants combined with their re-purposing and issues, we see the complexity of an information production process grow quite rapidly. It is no longer two-dimensional, as for each one of the production lines, but can be seen, rather, as a *non-linear* production network with *variants and their relations to each other* as the objects to be managed therein. Whilst it is easier to run parallel, specialized databases of the same information base for special functions, the consistency problems become exponentially difficult to handle as the databases and their complexity grow.

3.1 Motivation and Characterization of Variants

As described in the Section 2 "Terms and Definitions", a variant in our sense of the word can be an (ideally XML-structured) information object or a link object or a collection of them. Variants of the same variant set describe one and the same subject in a different way. The motivation for creating a new variant is triggered from criteria like new production channel, new user profile, new publication (of law, spin-off product), new variant in the product described (technical documentation), additional language. It is not triggered by production workflow issues which merely result in versions.

Typically, a variant is created by a *copy-edit-characterize* action on a given object performed by an editor who intellectually decides which information object (fragment) qualifies for a new variant, at what granularity a variant should be created, which content changes are necessary for the new purpose, and which are its characteristics/validity constraints. This new variant now has its own lifecycle, and can itself be the basis for a new variant.

The variants in a variant set are equally valid and useful in their specific context, according to their validity constraints. These constraints characterize each variant according to its occurrence – i.e. in which product, for which user group, at which point in time, it should be published. As these criteria are *information about information*, we call them *metadata*. In our model, both information objects and link objects can have metadata. These are an inherent part of the managed objects and are stored at version level. See Figures 1 and 2.

- A variant set is a set of managed objects making up all defined variants to a managed object. In the relationship of

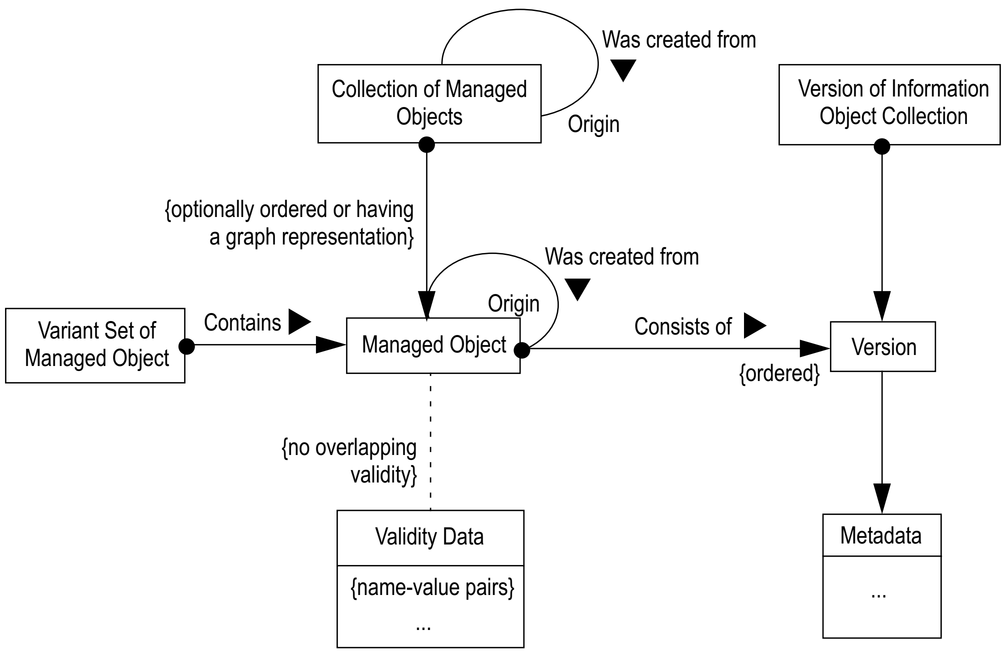


Figure 1: (XML-)Managed Objects, Links, Metadata and Link Metadata: the Building Blocks of Variant Handling (1 of 2)

variants to the variant set, validity data is needed to identify which variant is valid in which context. It is illegal to have variants within one variant set with overlapping validity. A given managed object can only take part in one variant set. All managed objects of a variant set belong to the same type of managed objects: either information object or link object.

- A collection of managed objects consists of managed information and link objects. The collection may have an internal structure (sequence or graph representation, e.g. container structure).
- A managed object consists of an ordered list of versions. There is never more than one current version available.
- A managed object has an optional association to another managed object of the same type indicating where it was created from by a copy operation.
- Metadata of managed objects are stored at version level.
- Link objects have relationships to versions of information objects. A link has a source and a target version. A link carries its metadata like information objects on version level.

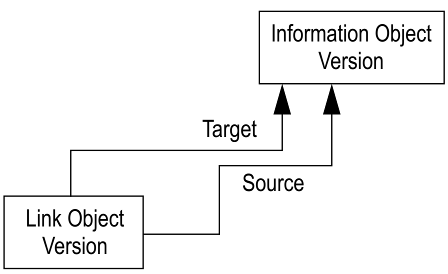


Figure 2: (XML-)Managed Objects, Links, Metadata and Link Metadata: the Building Blocks of Variant Handling (2 of 2)

With the given data model the following operations are possible:

- Finding the variant set in which a given information object is a member.
- Retrieving the matching information object for a given validity constraint with a given start information object (independently from the validity of the start information object)
- Traversing the origin chain of information objects.

3.2 What Happens Logically when Variants are Created

The following initial situation is given, as seen in Figure 3:

- Two information objects: IO1.1 and IO2.1
- Both information objects are linked by a link object LO3.1

A new variant of IO1.1 should be created now.The situation after generating the variant is shown in Figure 4.

Variant creation resulted in two new objects: a new information object which builds a new variant set together with IO1, and a new link LO5. The two links make up a variant set on their own.

In the next step a variant of IO4 will be created. The situation after creation is shown in Figure 5:

Since an information object can only be part in one variant set, the new information object will be part in the same variant set.

In the next step a variant will be added to IO2. The situation is shown in Figure 6.

Assuming we would like to create a publication with validity constraints that would fit for information objects IO1.1 in V1 and IO8.1 in V2. Remember that validity overlapping is not

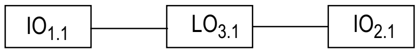


Figure 3: Creation of Variants: Initial Situation

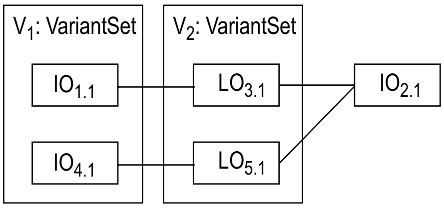


Figure 4: Creation of New Variants (1 of 4)

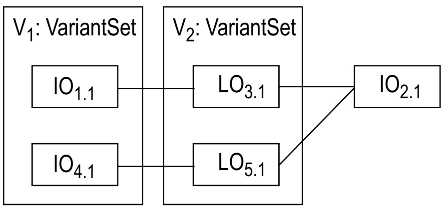


Figure 5: Creation of New Variants (2 of 4)

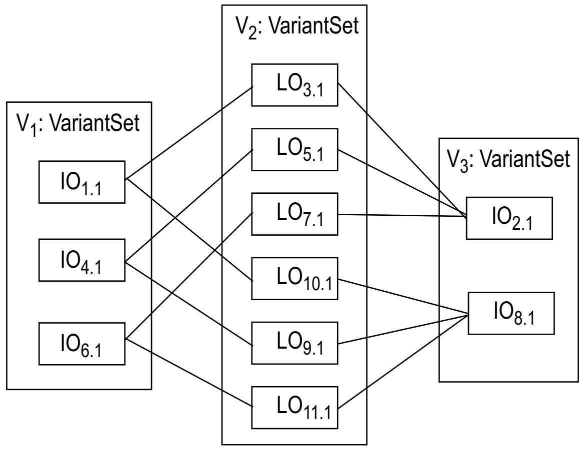


Figure 6: Creation of New Variants (3 of 4)

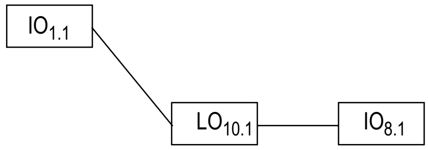


Figure 7: Creation of New Variants (4 of 4)

allowed so only one information object of each variant set will valid.

Figure 7 shows the resulting collection of information objects of the publication:

3.3 "Content-Managing" Variants on Object Level

A system supporting variant handling should provide efficient means for handling variants of managed objects (information objects and link objects). It must provide means to:

- Create a variant of an information object by copying it and adding it to the variant set (this may include creating additional links). The newly created object must store information about the information object it was copied from.
- Edit the validity data of a variant.
- Add an existing information object to a variant set.
- Remove an information object from a variant set
- Create a report on all variants within a variant set including their validity data
- Check for overlapping validities within a variant set with a given validity constraint
- Checking that at least one object within a variant set "is valid" according to a given validity constraint.

3.4 "Content-Managing" Collections of Managed Objects

Collections have editable metadata for classification and retrieval. Collections may have an internally defined sequence of their managed objects. Collections can be assembled by different methods:

- Searching
- Individual selection
- Link traversing
- Navigating views of managed objects (e.g. container structures)
- Combinations of the above.

The system should support bulk operations on the members of a collection:

- Editing metadata
- Setting workflow states
- Check-in and Check-out
- Moving and Copying
- Re-using
- Export
- Deleting

4 Serving Multiple Input and Output Channels

RW publishing is a typical information business that absolutely depends on user happiness with the *information* published. Their business is the *provision of* multiple existing texts and anticipating future expectations. *Information is the product*, either explicit or implicit (explicit = the texts; implicit = hyper/between/beneath-text).

4.1 A Reference Works Publisher's Real World...

Our reference works publisher (RWP) wants to market his content by generating various products from his database on different media for different target groups and by syndicating this database. The RWP partially generates the database himself and partially buys content from other publishers. He works with internal and external editors. The whole process is shown in Figure 8.

The RWP is embedded between suppliers and consumers of content. External resources for content are external authors and syndication suppliers. External authors write or review articles for given lists of lemmas. External syndication suppliers deliver whole databases of articles or assets (pictures) either on demand or based on their own content databases. The RWP also has internal authors. They basically have the same tasks as the external authors.

The RWP has several content management teams for different content building and maintenance tasks. Each team uses additional internal and external authors as well as syndication suppliers to perform these tasks. Each team has an editor in chief and about 3 to 6 team members.

For the actual building of *products*, there are several *product management teams*. They decide on the creation of new products, select content from the different databases, modify and adapt it as needed, produce the product via the production department and maintain the product probably over several years. They use several sales channels to sell the product (Figure 9).

RWP has basically two reference works content databases. The "medium reference works (RW)" and the "big RW". They evolved historically separated but the lists of lemmas has been synchronized in between, and there is an RWP wide list of lemmas. In addition to that, the chronicles content database is a database of historical articles for different topics (cities, politics, countries). RWP has a database of electronic pictures and an archive for multimedia assets. The archive is organized electronically but the different assets are partially only available offline due to expensive conversion costs and limited electronic storage capacity (films).

The products themselves are first of all a variety of reference works. There are small and big versions and special variants (e.g. physics) for special topics or target groups. The big version is also published on CD-ROM.

A separately edited multimedia version is published on CD-ROM. The web site publishes the "big RW" plus a lot of pictures and even multi media data. The web content additionally contains internal links for different lemmas. Users are even able to add their own content to different lemmas. This additional information is also stored in the web content database. By semi-automatic procedures, a topic map is maintained on the articles in the website and is interactively accessible for the user. The web editorial team also writes "actual articles" on actual topics. These actual articles are also stored in the web contents database.

4.2 Several Steps to Keep it Going

4.2.1 Project Goals

The main goal of dealing with variants, versions and configuration management in this project is to arrive at a *content centric* (as opposed to a product centric) content management. In order to achieve this it is necessary to *decouple* the creation of content from the formation of products and their production. This holds for the content structure as well as for the workflow.

4.2.2 Use Cases

Acquisition of information

1. *New or modified pieces of information pass through a product independent editorial process. Creation or modification has no direct effect on existing products.*
2. Additional or new information is to be acquired. An author is identified to (re)write an article.
3. Articles of a certain class (say all articles about German cities) need to be updated (e.g. a paragraph about their

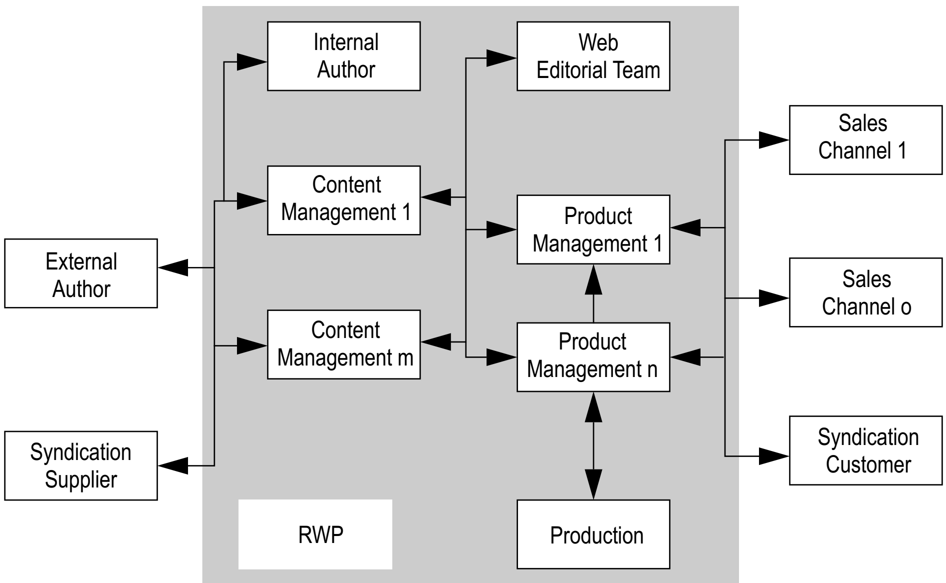


Figure 8: RW Production Environment

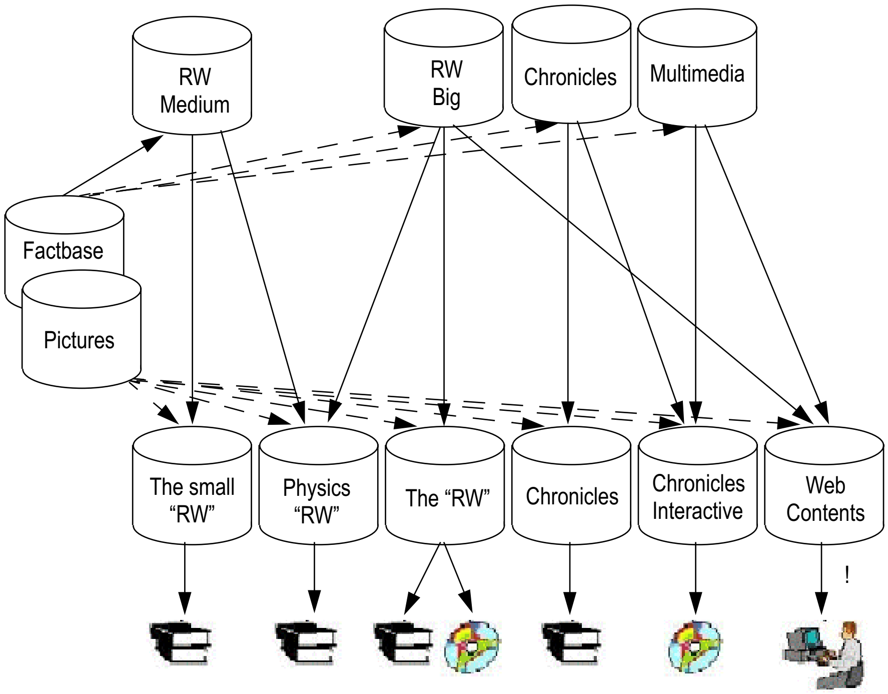


Figure 9: RW Publication Channels

breweries is to be added). All relevant articles are identified, either by selecting them manually or by retrieving them through a metadata search. The articles are passed through an edit / review process to ensure that all articles contain the required information.
4. New articles can be created and passed through the same edit / review process, if necessary.
5. If the group of articles intersects with another group which is already in a rework process, the collision has to be evaluated by an editor and the project may have to be postponed.
6. All articles of the class have successfully passed the editorial process. i.e. all articles contain the required information and have been reviewed. They are now ready to be used in any product.
7. Creation, modification (and even deletion) of information objects must not have any direct effect on existing products. This means the products must not adopt the modifications unless the responsible editor decides so.
8. *Remarks:* These "up-to-date" information objects are constantly changing at a rather quick pace.

Release of information

1. *When a class of articles has been modified successfully, the objects are made available for products (released). This released information is self contained and unchangeable except through overwriting by the next release.*
2. A class of articles has successfully passed a modification process (see use case 1: Acquisition of information).
3. The responsible editor acknowledges the successful acquisition of information and releases the corresponding class of articles. The relevant articles are identified manually or through a *metadata search*. The current content of all articles along with all links and designated metadata is set to a defined status to be the released version(s) of the information (read-only). A previous existing released version will be overwritten by the "new" release and the old, released version(s) will be discarded. All links between two information objects in the editorial process are translated into links in between the released versions of these information objects.
4. In some cases it was decided that objects are to be released automatically as soon as they had passed the editorial process of information acquisition. This can be regarded merely as a convenience improvement for the editors.
5. If a link cannot be established in the pool of released versions (i.e. one of the objects is not available yet) it will be ignored. However the link will automatically be established as soon as the missing released object is available (i.e. when the corresponding article is released).
6. The information is now available for the creation / modification of products. All relations (links) that are valid within the released information have been established. All designated metadata have been copied.
7. The pool of released objects must be self-contained. All elements (objects, links, metadata) must be updateable through releasing the corresponding objects again.
8. *Remarks:* The released objects basically form a (named) version. The one exception is, that the next release (i.e. next version) will overwrite the existing one.

Generation of Products

1. *An editor selects a set of information objects from the pool of released versions to form a product. Variants of these objects are made available exclusively for the processes of this product.*
2. *Precondition:* Released Versions of information objects available. The structure (storage, workflow etc.) for the product has been defined.
   The editor identifies all relevant articles for his product among the released versions. This is mostly done through metadata searches, as the collection is usually very big. The selection is stored in a so-called shopping cart (basically a storage of object identifiers in combination with the identification of the selection). A process is started to generate variants of the selected articles (and the links and metadata). The set of variants (the product) is self-con-

tained. The objects can be modified without any interference to other products.

3. *Post Condition:* All necessary objects and links are available for the product and can be modified.
   The product must be self contained and modifiable. All elements must be updateable through adopting the corresponding released version.
4. *Remark:* The product is derived as variants from released versions.

Maintaining Products

1. *Maintaining products covers local modification of information as well as updating / recovering information from the pool of released versions.*
2. *Precondition:* The product has been created and needs to be modified.
   In the course of processing the product along its special workflow (e.g. structural adjustments, proof reading, printing …) there is the need to modify the content. This can be done locally to adjust the content with respect to the publication media (i.e. to manipulate page breaks) or with respect to the designated audience. Another means of maintaining a product is to update the content from the pool of released versions. This can be split into two major possibilities:
   a. Product driven update: A set of articles is selected in the product (with the means of shopping carts). The objects in the set are updated from the pool of released versions. Optionally objects can be deleted if their correspondences are no longer available among the released objects.
   b. Released versions driven update: A new set of articles is selected in the pool of released objects (with the means of shopping cart). The objects inserted into the product or the corresponding product objects are updated.

   Note that a product can contain objects that have no correspondences in the pool of released versions. This can be the case if the objects were originally created in the product or if the corresponding released version has been deleted.
3. *Result:* The product remains a self-contained set of variants.
   All links that can be derived from links in the pool of released versions are established in the product.
4. *Remark:* The updating process can be seen as a creation of new variants in the product while the old variants are discarded.

5 Time Variants in Legal Publishing

The semantics of time plays a crucial role in the publishing of products based on legal documents: it is *the* criteria for the validity of the legal texts and references to them. If we talk about the laws themselves, we would rather think it is a versions issue we have to address. But in managing laws – and their fragments down to paragraphs – as information units, we think they must be treated as variants for validity reasons: the state of each production effective to a certain point in time (or time range) must be kept retrievable.

5.1 The Proof-of-concept Prototype at ZGR, Wolters Kluwer, Germany

The ZGR (central legislation editorial office) is a pilot sigmalink installation at Wolters Kluwer Germany. ZGR can be considered *the proof of concept project* with the following targets:

- enable editorial staff to use a CMS
- experience in CMS
- proof of general CM concepts
- proof of DTD concept
- get user feedback/usability issues
- explore basic customizing facilities
- find out about cost and efforts
- comparison to former way of working
- export system neutral data for further processing

Fundamental reasons for choosing the ZGR for the pilot were:

- Central law editorial is a complex task that gives a good example for solving challenging situations (esp. variant handling) with a CMS.
- Central law editorial is the “heart” of services a legal publisher in Germany may give to his customers; other data types (comments, decisions) may build up on this central base.
- There was a need for a high quality output format to be given to production.

The basic idea has been to build a data structure context “from heart to periphery”, beginning with central law editorial, going further with court decisions (which are referring to laws), and finishing with works of comment character, that may refer to everything else.

The implemented processes were shown at XML Europe 2002 in Barcelona, Spain.

5.1.1 Import

Existing data resources (a collection of law objects) are imported into the system through a defined import format. The format carries all *time-dependent information*, *as well as metadata and links*. This information is kept as elements and attributes in the SGML instances before importing them in the system. During the import:

- the documents are *split* into *fragments* according to structures defined in the DTD,
- status and other information about the information objects are extracted as *metadata* to be managed by the system and used as basis for searches and retrieval actions in the editing process,
- link information like *valid-from* is extracted as link metadata (see Section 5.1.2 “Variant Engine”.)

5.1.2 Variant Engine

The Variant Engine stores and manages all *time-dependent variants of a legal document* marked up in SGML/XML. Every time-dependent fragment carries a *valid-from* and a *valid-until* date. The date could either be a “real” date or a logical date like “to be defined by the government” or “valid until further notice”. Because the differences between the time versions of a legal document might be small – compared to the size of the document – the variant engine *reduces the size of the copied text fragments to a minimum*. Each fragment is under the control of the variant engine. The access to the variants for editing, reading, linking, and publishing is done through the engine. It also ensures that the *proper documents are assembled* from all the time-dependent text fragments stored in the database.

5.1.3 Extended Linking

Legal publishers sell added value to public laws. Maintaining legal texts is one task, but *assigning comments and connecting related texts with links* is another more valuable task. The link networks make use of a *common information pool* with all the legal texts. The publishers can create different *link layers or networks* over the same information object, like customer groups or specific purposes as the state of a legal situation ten years ago. Thus, it is an important precondition that *the links can be established without changing the data* and that they can be collected and managed as *different networks*. The system’s extended linking feature provides this functionality and is also integrated into the variant engine offering time dependent links.

5.1.4 Structure Search

Verbal cross references to legal documents make use of the strong hierarchy of the texts, e.g. *directive XYZ, §2, item 1, sentence 4*. Those laws which change other laws make heavy use of such references to identify the text portions to be changed, and it is one of the most important tasks for the publisher to keep track of all the changes in his law database. Thus, *easy following of these verbal references* is an essential requirement to the system.

5.1.5 Computer Aided Update (CAU)

The maintenance of laws means *applying all the changes from the changed laws into the publisher’s database*. This important but laborious task should be supported by the CAU. The changed laws have to be transferred into an electronic form which controls the automatic update program. Every in-between step can be checked manually before the automatic process continues. Thus, the CAU simplifies and speeds up the maintenance of law texts.

5.1.6 Collection and Publication

A published legal text might consist of a number of separate documents, decisions, comments, graphics, etc. connected to each other by links. This makes electronic publications on CD-ROMs or in the Web more useful than the printed versions and, moreover, if the links have metadata, they themselves can be treated as variant objects. The collection function of the system will *pack all valid information objects belonging to a specific publication with all their metadata and all their links into one publication*. The publication acts like an archive that can be extracted completely from the database, at any later point in time.

5.1.7 Export

The export process exports all the publication data from the database to the file system *for use in other tools* like typesetting engines or CD-ROM applications.

6 Conclusion

With information modelling concepts like XML plus complementary concepts for modelling meta-information like metadata and inherent knowledge structures expressed in the link objects, we have the methods we need for handling the constantly growing number of variants to meet the known and still unknown challenges in the information producing industries.

In our approach to variant handling, the heart is *smart content* in an abstract format which is independent of the tools it is produced with, media it will be published in, free from product specific information, thus being smart enough to serve them all in its varying occurrences that can be created on demand.

Note

This paper is based on a presentation given at XML Europe 2002 in Barcelona, Spain.

Acknowledgements

I would like to thank my colleagues, especially Dr. Martin Kreutzer, Johannes Müller, and Dr. Franz Weber, for their most valuable contributions concerning the logics of variant handling and Mr. Németh for his creative cooperation in the ZGR project and his consent for showing it as a use case in XML Europe 2002.

References

[Terris 2001]
John F. Terris: IT Program Manager, Allen Park, Valley Forge Technical Information Services, “Re-use, Re-purpose, Re-package: A General Engine Products, Inc. Case Study”, <http://www.idealliance.org/papers/xml2001papers/tm/WEB/04-01-04/04-01-04.htm>, XML 2001.

Business Maps: Topic Maps Go B2B!

Marc de Graauw

Interoperability between ontologies is a big, if not the single biggest, issue in B2B data exchange. For the near (and probably distant) future there will not be a single, unifying, widely accepted B2B vocabulary. Therefore we will need mappings between different ontologies. Since these mappings are inherently situational, and the context is very complex, we cannot expect computers to create more than a small part of those mappings. We need tools to leverage the intelligence of human business experts. We need portable, reusable and standardized mappings. Topic Maps are an excellent vehicle to provide those 'Business Maps'.

Keywords: Meaning, Philosophy, Semantics.

1 B2B Interoperability: the Problem

I will start with a quick summary of my presentation at XML2001 in Orlando, as this paper is a continuation of it.

We have lots of data and descriptions of those data. Take for instance the abundance of vocabularies for B2B exchange like xCBL, FinXML, FpML et cetera. Those vocabularies can be seen as ontologies. Older EDI technologies such as X.12 and EDIFACT are also ontologies. Beside those 'industry-strength' solutions, there are lots of tailor-made data exchanges between companies, often using nothing more than simple ASCII comma-separated files. Together with their documentation, those ASCII-files also constitute ontologies. And even within larger companies many different ontologies exist within the different legacy databases of the different departments. Those different data sources present huge interoperability problems.

One of those interoperability problems is finding out which data items from different sources are the same. To do that, we need to compare the meanings of those data items. This means we have to look up data definitions (metadata) for different data sources and compare those data definitions. Comparing human-made definitions is a tough job. Different companies and departments may come up with very different definitions for things that really are the same, and with very similar definitions for things that are very different in reality.

First of all, hard as we try, mistakes and obscurities occur in what we write down in our data definitions. Second, in making data definitions we may find that a lot of data are not that well defined to start with. In other words, when we make data definitions for a data source this sometimes is the first attempt to define the data at all, and when there already is a definition, it is often not precise enough. Third, when we make a definition like 'an employee is a person working at a company', we introduce many new words ('person', 'work', 'company') from natural language. When meanings in natural language are not precise, those definitions are not going to be precise either. We should just not think we can fix meanings once and for all in any but the most limited contexts.

2 The Mapping Model

In mapping B2B vocabularies we have to distinguish between type and token mappings. I use the linguistic terms 'type' and 'token' here; the IT terms 'class' and 'instance' or the mathematical 'set' and 'element' are very similar.

A token item in a B2B exchange is for instance:

```
<amount currency="USD">200.00</amount>
```

or:

```
USD;20000;
```

in some non-XML format.

The type of these tokens is amount. What matters is not so much the type itself, but the type description. An example of a type description is: "Amount is the total amount due excluding VAT for all goods ordered at webwarung.com". Additional type description data is: "amount is in USD, EUR, JPY or IDR" and "amount has a maximum of two decimals". Type description data are, of course, metadata. There is an important distinction between 'type' and 'set' or 'class' here. The mathematical notion of set (and the IT notion of 'class') presupposes an enumerable (though maybe infinite) number of elements. Common set notations are:

Marc de Graauw, born 1961, lives and works in Amsterdam, the Netherlands. He is an IT Consultant. He graduated cum laude in Philosophy from Nijmegen University in 1989. As a student of Philosophy, he studied Philosophy of Language and semantics in particular. His thesis was on the reference of nouns of tangibles, i.e. the relation between words such as "chair" or "cat" and actual chairs and cats. After university he started working in IT. In 1998 Marc continued working in IT as an independent consultant, specializing in intercompany data exchange, first in a traditional EDI environment, but soon in XML and Internet environments. Marc has published several articles (in Dutch and English) on XML and e-commerce and has been a speaker on conferences in the Netherlands. Recently he gave a presentation on XML2001 entitled 'What is "is"? Problems and solutions in mapping vocabularies'. His web page is at <http://www.marcdegraauw.com/> and he can be reached at <marc@marcdegraauw.com>

A = {1, 2, 3}

or:

B = { x e N | x > 3 }

In mathematics everything is necessarily well-defined. In natural language, things are not so clear-cut. There is nothing in the word 'love' which requires the existence of a fixed and well-defined set of occurrences of 'love'. Though the type-token distinction is similar to the set-element and class-instance distinction, type-token does not have the mathematical rigor of the former two. This is an important point, since the IT-tools used in B2B derive from the mathematical realm, and the business descriptions used in B2B derive from the natural language realm. So there is a clash of cultures here. The IT-side presupposes mathematical rigor, whereas the business side does not. Of course, the solution is simple: we have to make business descriptions more precise, so we can use them in algorithms. But note: this solution is the archetypical response from the IT-department, wholly from the mathematical, rigorous point of view. The business analyst might have a different solution: why do the IT-tools not allow more flexibility and less rigour. Or, as end-users usually phrase it: why does the system not do what I want it to do!

Back to mappings and conversions. A token-to-token conversion is in a B2B exchange context expressible in XSLT, or a programming language for non-XML domains. There is however no common tool to do type-to-type mappings with. There are not even many standards for type descriptions: usually data descriptions are distributed in some table-like format, usually containing things like name, data type and textual definition. However, nothing is guaranteed and the formats in which this is done are wildly different. The situation might improve somewhat through Dublin Core and similar initiatives, though Dublin Core is not specifically directed at this type of metadata.

Type-to-type mappings are the problem this paper addresses. On the type level we usually consider semantic and contextual information, whereas token mappings focus more on syntactical aspects of conversion.

3 Some Solutions and Why They Do Not Work

In this part I want to explore current solutions in XML and Knowledge Management. The first, which I shall call 'the naive approach', is: Let us make a new vocabulary which covers everything, then let everybody use that vocabulary. Probably everybody has thought of this sometime and found out it does not work in practice. Multiple vocabularies are a fact of life.

Another approach to interoperability is the use of Published Subject Indicators (PSI) as used in Topic Maps. The basic idea is to make public libraries of unique IDs for things. In our vocabularies we incorporate PSIs, and then we can compare the terms in our vocabularies. In an informal example:

Topic: 'United States of America'; PSI: US

Topic: 'Verenigde Staten van America'; PSI: US

The PSIs in the English and Dutch topics allow us to conclude that both topics are the same. Note that this really just shifts the problem from vocabularies to public libraries. In general we can say this approach is successful if the problem space consists of clearly delimited entities and there is a widely accepted canonical public library. Examples of areas where this approach will work are for instance ISO currency and country codes.

Now back to a 'real world' example. In actual mappings between ontologies, we often do not really establish semantic equivalence in a true sense as needed in PSIs. When we have found we can use 'CustomerAddress' of our trading partner as the 'billing_address' in our online billing application, we stop. We do not need to find out whether they are truly equivalent in all circumstances. There is no direct business need to find out whether they are equivalent in all situations, and therefore the boss will not pay to find this out. Solutions like PSIs do not work here, because PSIs require true semantic equivalence. The interesting observation here is that most real world mappings are unidirectional: we translate from a source ontology to a destination ontology for a specific business process. For instance, an order goes from buyer to supplier. It does not go back (though a different document such as an invoice or order confirmation might go back). So for an order only a translation from the buyer's ontology to the supplier's ontology is needed. This unidirectional nature of business exchange means that often we do not establish equivalence relationships, but subset relationships between ontologies. In the above example, 'CustomerAddress' is a subset of 'billing_address'. All instances of 'CustomerAddress' constitute a valid instance of 'billing_address'. We do not know whether the reverse is true, and in this example we do not need to know either.

It might be tempting to conclude that we simply have to make a mapping between every two ontologies we use. That, however, is going too far. Even when we do not always establish true semantic equivalence relationships, the mappings we make are certainly for a great part reusable. What we need to do is capture knowledge about the mapping process itself. We need to store the fact that we can use 'CustomerAddress' as 'billing_address' in this particular context. Then, when someone else needs to find out whether 'CustomerAddress' can be used as 'InvoiceAddress' in a different context, they can use this information. When we store this kind of information, we could facilitate the process of mapping ontologies through the use of semi-automated tools which show existing mappings for items in our ontology that we need to map onto another ontology. The human expert making the mapping can still make all the relevant choices and provide new mappings where existing ones cannot be reused. Such semi-automated tools could then generate a new mapping, which also can be stored to provide information for the next one. It would also become much easier to exchange information about mappings without having to provide full one-on-one equivalence relationships.

4 The Elusive Automatic Mapper

A lot of the research effort in this area goes to automatic mapping. The basic idea is (usually) to describe data on a type level in such detail at design time that new mappings can be established automatically at run-time. Note that this is an enormously complicated task. It is an important one too: I think it will take some time before we can utter the next sentence happily: "Hey, that is funny, yesterday my computer sent $50.000 to a company I have never heard of!" Or to put it otherwise: I do not want my computer to do that yet!

Let us look at an example to see how complex the task can get. Best Insurance and CarCovered.Com are merging. Best Insurance does not sell car insurance, so there is a huge chance for cross-sales of car insurance to the customer base of Best Insurance. We want to use Best's customers as prospects for new sales, but we do not want to send offers for car insurance to Best customers who - coincidentally - already have CarCovered.Com insurance: that makes us look real stupid. Questions a marketing manager will ask in such a situation are:

- Do we expect overlap in customer base? Is there regional or industrial overlap?
- Can we find most of the doubles? (With all the different name spellings, double addresses, PO boxes versus regular addresses etc.)
- How bad is it if we overlap? How much effort do we put in finding overlaps? Is 2% remaining existing customers acceptable? Or 0.2%?
- How much money does it cost to remove double addresses? How stupid do we look when we make mistakes?
- etc. etc.

Only then can the marketing manager decide whether Best's customer data are actually usable as prospect data for CarCovered.Com. So how could a computer decide whether 'Customer' can map onto 'Prospect'? This takes a whole new level of AI. Of course, I have chosen a really hard example, and there are other simpler contexts. My personal experience as a B2B consultant however is that things usually appear simpler than they are in B2B data exchanges.

Of course it is fine in research to tackle problems which are hard, and may not have a solution in the near future. For the time being that is where automatic mapping will stay: in the lab. What is needed in the short to medium term is not so much automatic mapping but tools to leverage human brains: tools which enable business analysts to make new type-to-type mappings quickly and reliably, using human intelligence.

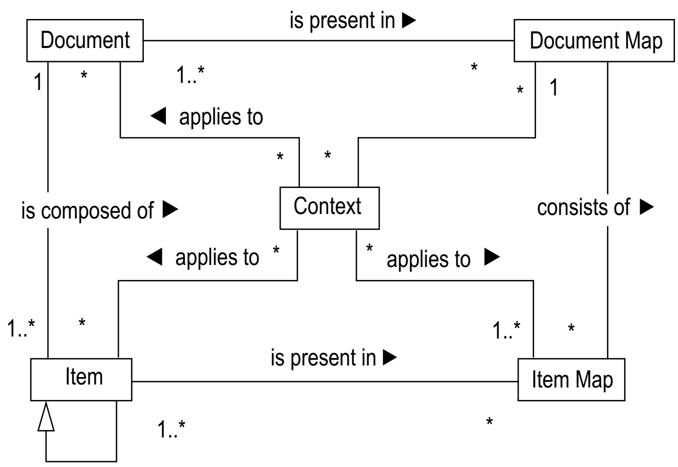


Figure 1: Incorporating Knowledge to Maps

5 The Knowledge in Mappings

This model shows what is necessary to store when we want to store the knowledge in mappings. First we must distinguish the documents which are exchanged in B2B exchanges and the items which make up those documents. The precise structure of the document can be stored as a DTD or Schema in the document class if necessary. Items can belong to domains (data types) which in turn can be specializations of other domains. In the left-hand classes we would store the information on the business documents in use by relevant business partners. There is no need to store the full definitions of the data; it is sufficient to store identifiers that uniquely identify the documents, items therein, and domains in use. After all, the full definitions are probably already stored somewhere; simply copying them would only introduce unnecessary redundancy.

On the right-hand side of the model we have the actual mappings. First, the fact that some (usually two) documents are mapped onto each other is stored in the class document-mapping. The document-mapping is related to one or more item-mappings. The item-mappings store not only the identifiers of the mapped items, but also the kind of mapping: is this an equivalence relationship, which is potentially bidirectional? Or is it a unidirectional mapping, which means it is at least a subset-superset relationship? We can also store conversions. For instance, if the destination document allows last names of only 25 characters, but the source allows last names of indefinite length, we could specify that names need to be truncated to 25 characters, or that this is an error. We could also store more complex transformations. They should, however, be readable for ordinary humans, which would rule out XSLT 'as is'. The intended users of a tool based on this model are business analysts, not XML programmers. (An intelligent tool could of course store XSLT for a large class of transformations and show the results in natural language.) Last, we can store domain conversions, so we would not have to store the same YYYYMMDD to DD-MM-YYYY conversion for every date in the document.

The central class is context. This is context in the broadest sense; this class could store information on B2B vocabulary, region, country, company, business unit, timeframe and whatever is necessary for the mapping under scrutiny. Context would apply to all other classes. If unspecified, an item mapping would usually inherit its context from the document mapping in which it is contained. If the document mapping is concerned with mapping some Invoice to some billing_doc, the item mappings will be valid only for this mapping unless specified otherwise. However, it does make perfect sense to store context for item mappings. We could for instance store that the 'last_name' of trading partner one always maps onto 'GivenName' of trading partner two. Context also applies to the left-hand side, though maybe we would only want to store the trading partner involved here. Of course, the Context Drivers of ebXML would constitute a good starting point for defining context.

6 Business Maps

The data model maps quite nicely onto core Topic Map constructs, to yield what I will call 'Business Maps'.

What is more, Topic Maps offer the facility to merge two distinct Topic Maps. This is an excellent way to compare separate, portable B2B-mappings. When we have two Business Maps, say one from Sales Europe and one from Sales Asia, and we want to make a new Map for Sales South America, we can merge the existing maps. Any references to external message items will be merged when they have the same subject indicator. Fortunately it is relatively easy to establish stable and unique subject indicators for vocabulary items: one can assume that the vocabularies act as namespaces, i.e. each item in it has an unique name, and one can assume the vocabularies to be identifiable by a URI. Suffixing the item name to the URI will do the job of establishing unique subject indicators for vocabulary items. (There are other ways to merge topics bases on names, but this is the most robust and appropriate one in this case.)

Once the Business Maps are merged, we can use the scope to filter out the business processes we are interested in. So Business Maps can provide an easy and flexible way to reuse knowledge stored in mappings: they are portable, reusable mappings come true.

Quite a few things remain for this vision to come true. The best thing of course would be an accepted standard for Business Maps. Having this, we would be able to exchange mappings with all companies using this standard. Note that this is a far less ambitious and more tenable goal than establishing a single unifying B2B ontology. This approach could also prove to be a viable way to achieve inter-company interoperability, still a big problem in the world of ever-merging large companies... We would need tools to support the Business Maps – querying, filtering, im- and exporting, and creating and editing them. And possibly we would need a description of the properties of applications processing Business Maps. The Topic Map standards do not say a lot about what scope means (intentionally so, this is up to the applications). A description of what 'context' means in this model would therefore be appropriate. All in all, this model could provide for a huge facilitation of human-mediated ontology mapping.

Mapping Model	Example	Topic Map construct
document	Invoice	topic
item	CustomerName	topic
document-to-document mapping	Invoice maps to bill	association
item-to-item mapping	CustomerName maps to company_name	association
context	Vocabulary, Company, Region, Industry, ...	scope
external document description	www.bizwords.org\invoice	occurrence (role: business document description)
external item definition	www.bizwords.org\amount	occurrence (role: definition)
external item datatype	www.bizwords.org\date	occurrence (role: datatype)
external item example	www.bizwords.org\amount\example	occurrence (role: example)
vocabulary identifier + document identifier or item identifier	www.bizwords.org\amount	subject identity

Table 1: Business Maps Model

7 Conclusion

Interoperability between ontologies is one of the (if not the) most important problems in B2B data exchange. For the time being, making mappings will mainly be a human job. Therefore we need a way to leverage human intelligence to make all the required B2B mappings. Portable, reusable mappings would accomplish this. Those mappings would need to store information on business document mappings and the context that applies to those mappings. Topic Maps are an excellent vehicle to store such information, thus yielding "Business Maps".

Samples of Business Maps are available at <http:/www.marcdegraauw.com/itm>.

Note

This paper is based on a presentation given at XML Europe 2002 in Barcelona, Spain.

XML and Word and XML: Downcasting and Upcasting XML Documents

Stefan Hermann

Microsoft Word (MS Word) is the most commonly used document preparation system today that is available on nearly every desktop computer. As such, MS Word is typically not only deployed as a type-writer like, easy-to-use document editor but also, despite a lot of prejudice, as a capable document formatting-engine that suffices for many applications. However, it lacks direct XML support in both usage scenarios. A primary drawback seems to be its WYSIWYG based user-interface paradigm that does not force users to structure their documents logically while creating them. Simply type and format – that's the typical way Word documents are prepared, leading often to unstructured accumulations of text that may be printed – but that's about all. However, this WYSIWYG paradigm might be the simple admission ticket for many typical MS Word users into the world of XML documents. By strictly following some simple rules, they may prepare – with the help of their accustomed document editor – documents that may be translated automatically and tool-based into XML of reasonably good quality. Another major drawback when using MS Word seems to be its resistance to importing documents in any open format e.g. structured documents like XML. MS Word accepts most proprietary formats as input but remains closed to any new requirements arising in the application domain of XML and its accompanying standards. This paper describes in general the basic architecture needed to embed MS Word into an environment enabling the import, export and roundtrip of XML documents. In particular it describes the two tools, downCast and upCast, that implement such an XML architecture for MS Word based on the additional standards XSL and CSS.

Keywords: CALS, CSS, DocBook, Downcast, DTD, HTML, Microsoft Word, RTF, Schema, Unicode, Upcast, VBA, XHTML, XML, XSL.

1 Documents

Before we look into how Microsoft Word (MS Word) can be embedded as an XML-enabled tool into an XML environment, let's have a brief look at both the document and its processing and introduce some terminology. Note that, for clarity, facts and procedures not crucial for to the goal of this paper have been simplified.

1.1 Definitions

There are many points of view from which a document may be viewed and classified. Traditionally, a document is a thing describing something. We can hold it like a newspaper or letter, and we can store it, on a library shelf, filing cabinet or digitally

To establish a clear and precise terminology for the rest of this article let us define initially that in a document preparation process the term *intended document* qualifies the information someone has in mind (metaphorically speaking: in his head) when creating a document. We further define the pure information that stands behind an intended document the *abstract information* that consists of *logical objects* connected by a *logical structure*.

Analogous to the term *intended document* the term *perceived document* is used to denote the information someone has in mind after he has comprehended a document. Again, as for *intended documents*, we call the pure information that stands behind a perceived document *abstract information*.

The term *concrete document* shall denote the physical representation of a document, something one may (at least nearly) touch, and we call the information that is contained in a c*oncrete document*, *concrete information*. Let us assume that an instance of a concrete document may either be a printed thing e.g. an article consisting of glyphs arranged left-to-right with applied line-breaking and page-breaking printed on sheets of

Stefan Hermann is one of the founders of infinity-loop GmbH <http://www.infinity-loop.de/>, where he currently serves as CEO. He received his Masters degree in computer science and his doctorate from the Technical University of Munich, Germany, in 1995 and 2001, respectively. His doctorate focused on design specification methods for structured documents in the digital publication process. In 1997 and 1998 he was a visiting researcher at the HKUST, Hong Kong, working with Prof. Derick Wood. Currently he is a visiting lecturer at the Technical University of Munich, focusing on XML based document processing.
<hermann@infinity-loop.de>

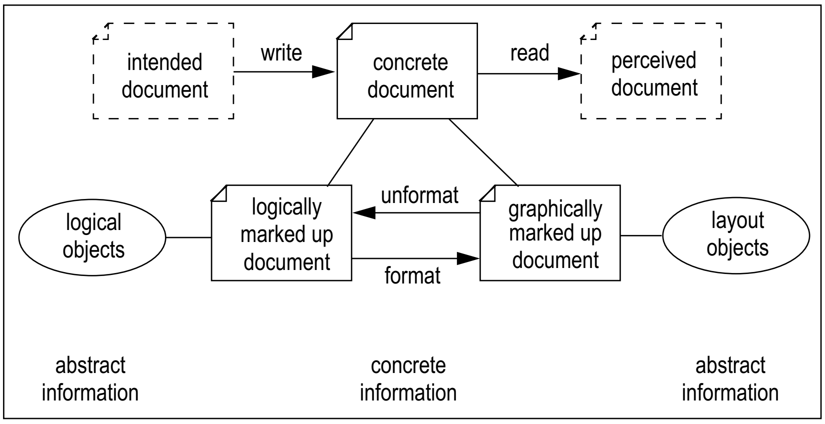


Figure 1: Processing Steps

paper or a sequence of bar codes or even morse codes applied on paper strips. Or it may be something digital.

In this last case there is a range, from *final-products* (like pixmap and PostScript or PDF documents) where (in a simplified view) all calculations needed to display or print a document are already carried out, to still changeable/editable *non-final-products* like MS Word or XML documents, which require some more processing before we can touch them.

In order to refer to specific aspects of concrete documents that are of interest in this paper, electronic concrete documents like XML documents are defined as *logically marked up documents* and electronic concrete documents that exist in a document format like MS Word are defined as *graphically marked up documents*.

Further we define that logically marked up documents consist of logical objects connected by a logical structure and that graphically marked up documents consist of *layout objects* (objects available in MS Word used to create documents, e.g. characters, paragraphs, lists, tables) connected in a *layout structure*.

1.2 Processing Steps

Now that we have introduced the terminology we may analyse the central steps in this process (see Figure 1.)

1.2.1. Concepts Behind Writing and Formatting

When someone writes a logically marked up document (e.g. an XML document) on a computer system he may do so by using a plain text editor. This kind of editor offers a rather simple view of the created document: displaying the document as a flat stream of characters containing both the markup language and the data of the document.

In order to get the written XML document displayed in a more humanly accessible way we have to apply a formatting *process* that transforms the logically marked up document into a graphically marked up document. The principal concept behind this formatting step is quite simple: we specify how single logical objects should be transformed into the formatted output that consists, as defined, of layout objects. In order to assign visual semantics to the logical objects, these layout objects may both offer parameters to individualize the applied formatting and they may be combined with other layout objects. In simple cases (that we assume to represent the majority of reasonable cases), the logical structure of a document will be structurally and functionally equivalent to its layout structure.

Someone using a tool like MS Word to write a document is creating a graphically marked up document consisting directly of layout objects rather than a logically marked up document. In such a case, there is no need to map logical objects onto layout objects. But, be aware that, as a consequence, the person who writes the document may create the logical structure of the document only indirectly, by means of applying the formatting that implies the desired logical function. MS Word does offer some mechanisms to apply named formatting decisions – via a style sheet approach that offers users the ability to change the formatting of equivalent parts in a document (e.g. all headings of level 1) by only changing the definition of a style rule.

Worth citing here is the fact that the concept of layout objects is mentioned and introduced in many different publications, like the DSSSL (flow objects) and the XSL standard (formatting objects).

Probably the most interesting paper that introduced the concept of layout objects preceding all other publications and which is essential reading, was written by M. Murata and K. Hayashi and goes back to ideas first described and published by P. Pedersen (hierarchies of formatters) in conjunction with the ODA standard.

1.2.2 Concepts Behind Reading and Unformatting

Someone intending to comprehend a concrete document must first read it. He processes layout objects displayed on some medium that are perhaps combined with other layout objects, and he assigns (based on knowledge he acquired while learning to read combined with other knowledge) semantics to these objects. In doing so he synthesizes the perceived document and restores a logical structure that may or may not be consistent with the one that was intended by the concrete document or intended by the document's author who had in mind his intended document.

In order to restore the logically marked up document from a graphically marked up document (no matter whether this logically marked up document existed or not) one has to perform a processing step that may be called unformatting. Details concerning this step are described in the next chapter.

2 upCast

The most commonly used document preparation system today that is available on nearly every desktop computer is MS Word. The typical way in which MS Word is deployed is a typewriter like usage: simply type and format. Thanks to the WYSIWYG based user interface paradigm that does not force its

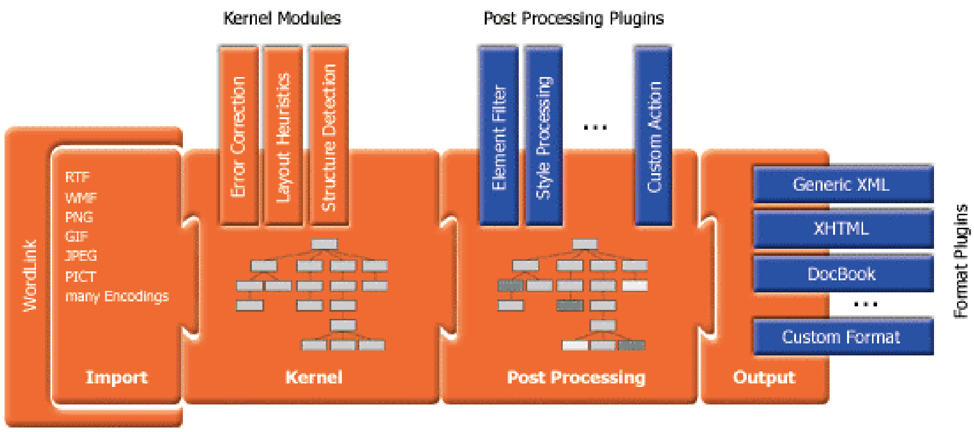


Figure 2: Upcasting Process

users to logically structure their documents during creation, documents created with MS Word may easily feature a poor logical structure (in the underlying data structure of the created documents) while still looking good (featuring a perfect logical structure) in print.

MS Word is easily used to create documents that feature a reasonably good logical structure as required in many usage scenarios that occur today. MS Word and its underlying document format RTF (Rich Text Format, a document format standard defined by Microsoft) to a certain extent support the logical markup of documents with the concept of styles for e.g. sections, paragraphs and characters that (we assume this in good faith) are easily usable even by unskilled users. MS Word also offers built-in and dedicated support for advanced features like tables, lists and footnotes in order to create pleasingly structured documents.

Thus, the WYSIWYG paradigm of MS Word could be the simple admission ticket into the world of XML for many typical MS Word users. Strictly following some simple rules, they may prepare – with the help of their accustomed document editor – logically structured documents.

Unfortunately, MS Word offers no native support of XML.

See an overall view of the upcasting process in Figure 2.

2.1 Upcasting: Conversion from MS Word to XML

In order to embed MS Word into an environment enabling the generation of XML documents (or to import Word documents into any available XML editor), one needs an external tool that can convert these documents into XML. We call this direction of document conversion upcasting (put simply, *deformatting*), leading directly to the name of our product: upCast.

From a purely functional point of view upCast is a standalone (usable independently from MS Word) RTF to XML converter that is implemented in Java. Our marketers remind us that Java implementation allows upCast to convert all documents created in applications that offer RTF export functionality, not just MS Word.

Technically, upCast is a quite complex system based on ideas developed at the TU München during the last eight years. Development on a commercial basis began in 1999 by a small team of programmers previously specialising in document formatters for structured documents.

The aims we set ourselves for the development of upCast were quite simple. First of all, we had to implement all the knowledge we gained while working at the TU München in the area of document processing and formatting. Simultaneously, it was obvious that we had to get as much real world experience as possible with our product in order to steer our development into the right direction (thus we offer a free unlimited version for private and non-commercial use). We decided that our implementation had to be purely standards-based wherever possible. Java was chosen for the implementation language due to its system independence and object orientation enabling us to use modern design principles for upCast's architecture. Today upCast also offers C/C++ and Visual Basic interfaces enabling upCast to be embedded in any product that needs RTF to XML conversion functionality.

upCast was designed to work with most different software architectures found in the area of document management systems by offering a family of products based on the upCast kernel implementing the core conversion functionality. Today, upCast fits a spectrum of scenarios from the single independent workplace application to complex workflow processes running on company wide server based installations.

In order to convert an RTF document, the upCast system first imports the given document from different possible sources (e.g. files or network streams). While doing so, the given logical structure that is directly specified in the document by means of RTF constructs (via the style sheet and some other constructs offered by MS Word) is analysed and used to construct an intermediate document tree reflecting the logical structure collected during capture.

The very important second step ensures the quality of the generated XML, upCast applies constantly updated heuristics to the intermediate document tree in order to infer and create a logical structure that matches the intended logical structure of the source document author as closely as possible. This step is absolutely necessary due to the way MS Word (and RTF in particular) works.

MS Word's WYSIWYG GUI approach and the fact that documents processed in Word are graphically marked up may lead to two different classes of problems. The first class is caused by artefacts in Word documents not perceivable by the author at the time of writing. A typical example is empty (and thus invisible) paragraphs – e.g. the author may have actually completed a list and begun a new one but the perceived document displayed on the screen may suggest to the eye that there is only a single continuous list.

The second class of problems is caused by the freedom users have in MS Word while graphically marking up documents. For example: due to its formatting an item in a list may appear to be a sequence of paragraphs: all paragraphs of the item are indented by exactly the same amount as the first paragraph – MS Word, however, ends the item, and the list, after the first paragraph of the item due to insufficient data structures offered directly by MS Word.

Having improved the logical structure in this second step, upCast may be required to perform some post processing steps to transform the logical structure created so far to incorporate knowledge that is present only in a certain application environment for a specific customer. The structure created by this step is an enriched logical structure that reflects all the information needs existing in that specific environment. The processing for this step may either be implemented in XSL or in Java, depending on the complexity of the respective requirements.

upCast relies heavily on the fact that the way authors graphically mark up their documents (using standards on how to format certain document parts that are acquired while learning to write) implies a very close 1:1 relationship between the layout structure of a given document and the logical structure of the same document – otherwise no one could read and understand a document somebody else has written. For this reason upCast contains a fixed, specific standard DTD that may reflect all possible logical structures for documents created in MS Word. Mathematically, all possible DTDs behind documents created with MS Word are simply equivalence classes of upCast's own DTD. There always exists a function to map documents marked up in upCast's DTD into one of these other equivalence DTDs.

Additional features of upCast include:

- Full recreation of the logical document structure with automatic section nesting and support for parts.
- Paragraph and character styles are processed.
- upCast offers powerful table translation, including row and column spans, offering support for both the HTML and CALS table model.
- Nesting tables are processed properly.
- Footnotes, hyperlinks and references are converted to elements, page headers and page footers are handled, document properties (including user-defined properties) are processed.
- Fields and nested fields are supported.
- Embedded WMF images may be converted into a pixel format (jpeg, png, bmp or pict). All images may be scaled to the output resolution of the target platform.
- upCast deals with any combination of nested lists, tables and any other combination of layout elements that might occur in MS Word documents.
- upCast fully supports Unicode and UTF encoding of XML documents.
- On Microsoft Windows systems upCast may directly convert MS Word binary files if MS Word is installed.

upCast offers the direct export of MS Word files to XHTML (content or layout centred), CSS, Docbook and upCast's own DTD or Schema (content or layout centred).

upCast may be customised to easily support many DTDs as used in real world applications.

3 downCast

As XML slowly becomes an accepted standard format for document-centric applications in the industry and thus more and more XML documents are available (often as the only provided format) the question arises, how to print or further process these XML documents. Again, MS Word is available on most computers and would be a good choice, but unfortunately it offers no native support for the import of XML.

Thus, in order to embed MS Word into an environment enabling the print-output or editing functionality of XML documents, one needs an enabling tool and architecture to convert XML documents into a format which can be read, processed and exported by MS Word. RTF is a suitable import tool, offering and supporting all the features that are available within MS Word.

3.1 Downcasting: Conversion from XML to Word

Despite a lot of prejudice, MS Word is a capable document formatting engine that is acceptable for many applications that need to print processed documents (either in print or on screen) containing simple and sophisticated layouts. So, it's a good idea to use it. if we can get XML files into it.

The approach we have chosen is based on some simple requirements. Firstly, it should – of course – be standards-based as far as possible but at the same time, technologically as reasoned and application oriented as is necessary to get good results in a reasonable way. Thus, we discarded the obvious XSL-FO solution (supplying us with a general vocabulary describing a formatting task) and instead, rely on the DTD developed for upCast (in order to get a specific vocabulary adapted to MS Word describing the desired formatting). As explained in the upCast Chapter it can be seen that the upCast DTD reflects both the logical and the layout structure of documents written in MS Word.

For the general conversion process from XML to MS Word, downCast relies on XML documents that are valid according to the upCast DTD. For specifying layout properties, we have chosen a CSS like style property approach (actually a CSS-2 subset, with added proprietary properties where necessary). This lets the user specify name-value pairs in order to achieve the exact formatting of XML documents into MS Word.

3.2 Roundtrip: XML and Word and XML

When trying to use MS Word as a two-way or roundtrip XML editor (import and export of XML documents), the requirements for the import process and thus the tool used for conversion of XML documents to MS Word gets still more challenging than described above. We have to ensure that the logical structure of the document imported to Word is congruent with the one written on export to XML – otherwise this exercise will fail.

For roundtripping XML documents through MS Word, we determined that it is essential not only to convert the logical structure present in the source XML document to a visually equivalent representation, but also to use the structuring mechanisms supported natively in RTF as completely as possible. Only then would we succeed in retaining that logical structure without unnecessary loss of information when exporting to XML. Otherwise, the clues available in the document to be exported in the upcasting process would be about as useful as those available in an OCR process – and we all know about the problems encountered there.

We are content that we have met the roundtrip processing requirements. The decision to use upCast's DTD as the basis for the development of downCast was correct. As we have seen, importing a document into MS Word with a possible need to export some time later demands functionality that may not be supplied by a general purpose standard like XSL-FO. In addition, the selection of CSS to steer details of the desired layout is obvious due to its acceptance and user-friendliness.

4 Conclusion

We believe that our design decisions on the architecture of upCast and downCast is have made the products well suited for many applications, both now, and in the future, where MS Word needs to be used in an XML-enabled environment. Some ten-thousand installations of upCast world-wide demonstrate its usability from small projects where only a few persons are working on XML documents, up to projects where upCast is integrated in a central client/server software with hundreds of users accessing its functionality. Our latest product, downCast, is already in use with some pilot applications and is closing the import-gap that still prevents the use of MS Word in a huge number of projects that need both import and export of XML documents.

upCast and downCast can be implemented as stand-alone tools that may be used independently of MS Word on any Java-enabled platform or directly integrated into Microsoft Word. This choice gives our customers the freedom to implement XML to Word to XML functionality in any architecture they need in order to solve their content management needs.

We hope that upCast and downCast are useful in promoting the idea of XML in a current world of non-XML tools. We thank all the people who constantly provided such excellent, and helpful, feedback during the development and refinement of our products over the past years.

Note

This paper is based on a presentation given at XML Europe 2002 in Barcelona, Spain.