XML-Based Software Development

Baltasar Fernández-Manjón, Alfredo Fernández-Valmayor, Antonio Navarro, José Luis Sierra
Grupo de Investigación en Ingeniería del Software e Inteligencia Artificial.
Departamento de Sistemas Informáticos y Programación
Facultad de Informática, Universidad Complutense de Madrid
Avd. Complutense S/N. Madrid 28040. SPAIN
{balta, alfredo, anavarro, jlsierra}@sip.ucm.es

Abstract: 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[1] [16] 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[2] or XML *Schema* [22]). XML does not concern about 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[3] [4], the predecessor of XML, was formulated to standardize the languages used in areas of application such as avionics or defense systems. In these applications the three former requirements of use are a primordial 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 [8]. HyTime was developed as an SGML extension to facilitate the description of hypermedia applications. HyTime demonstrated that marked documents could be used to describe the complexities of a non-trivial software application. In addition, other languages such as DSSSL[4] [7] were provided to deal with the processing of SGML documents. Most of these early initiatives have been reinterpreted in the XML world. The analogous to HyTime are several XML-based languages that cover different aspects of a

[1] eXtensible Markup Language.
[2] Document Type Definition.
[3] Standard Generalized Markup Language.
[4] Document Style Semantics and Specification Language.

hypermedia application. SMIL[5] [18] is a language or *application*[6] that facilitates the specification of the timing and synchronization in a multimedia presentation. XLink/XPointer [19][21] are linguistic patterns used to describe different types of hyperlinks and anchors between documents. XSL[7] (the XML-reinterpretation of DSSSL) [17][20][23] introduces the notion of style sheet formed by a set of transformation rules that specify how to transform XML documents into 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 [3] motivated our interest in SGML. Using Dexter model precepts [5], 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 susceptible of 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.

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[7] [2]. 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.

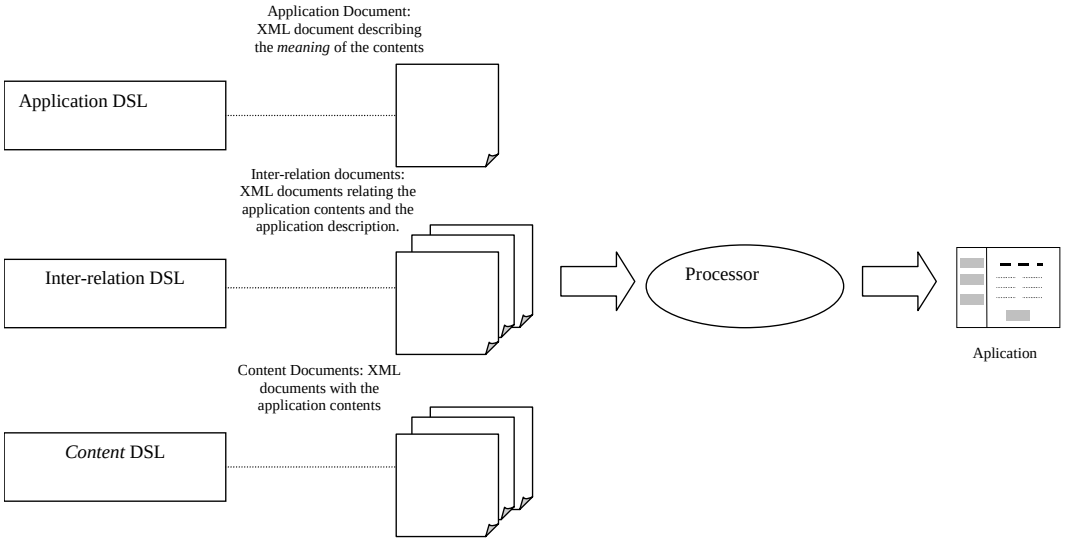

Fig. 1. Using XML in a DSL based approach to software development.

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

[5] Synchronized Multimedia Integration Language
[6] In the XML context, *application* means a XML based markup language.
[7] Domain Specific Languages.

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 possible to express the relation between content and application documents using XML standardized languages such XPath [20] or XSLT [23].

The construction of a specific application (Fig. 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.

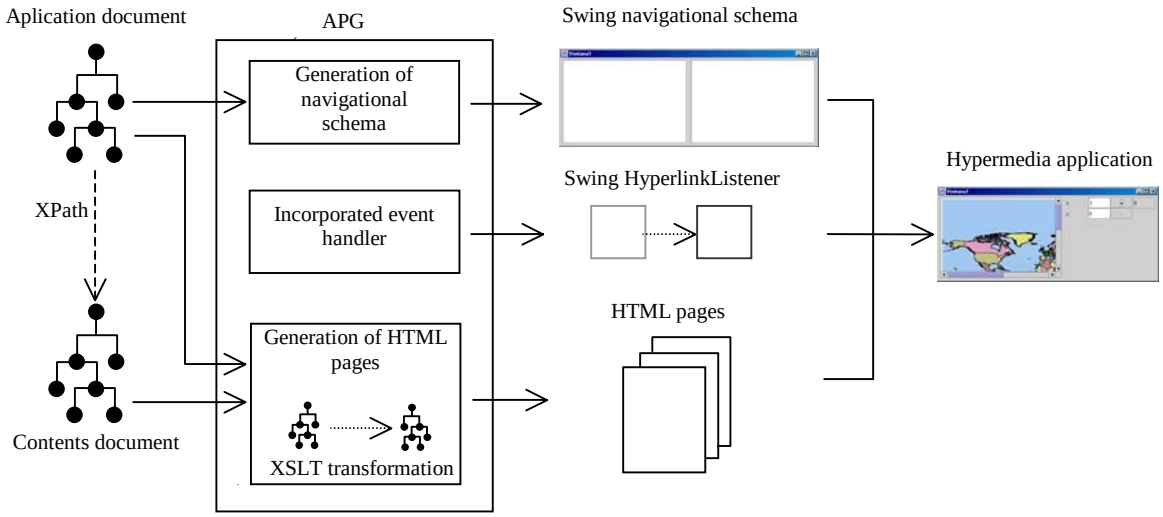

Fig. 2. APG working scheme

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 [12]. Starting with the classic hypermedia models and making a detailed domain analysis (specially 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 to represent, 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 means to characterize the user interface and the navigational relations between elements of the interface. The third are the *canalization functions*. These functions provide a mean to relate the contents graph with 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 forth element of our model is the *navigational semantics*, that characterize the response of the hypermedia application when the user selects a link. Finally, there is the *presentation semantics* that describe the most specific characteristics of the application such as font size, family and color.

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 relations 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 at 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 [10]. 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 all the process previously described only covers the conceptualization and prototyping phases of development [11]. We are considering to extend APG to cover the whole process of hypermedia development following an approach close to the one described by the Amsterdam model [6] and the language SMIL. Fig. 2 sketches how APG works. Fig 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= "p1.1" size=" 140, 320">
       <paneContent>
        <defaultContent>/contents/index</defaultContent>
       </paneContent>
       <linksDestination pane="p1.2"/>
     </pane>
     <pane id= "p1.2" size="320, 200">
       <paneContent>
        <groupContent>/contents/subIndexes</groupContent>
       </paneContent>
       <linksDestination pane="p1.3"/>
     </pane>
     <pane id= "p1.3" size="400, 500">
      <paneContent>
       <groupContent>/contents/apContents</groupContent>
      </paneContent>
     </pane>
   </window>
</application>
```

Fig. 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 Fig. 1. To build this framework, instead to start 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 [13][14]. We call this approach DTC[8].

DTC uses XSLT as the inter-relation language and associates a software component with each of the elements of the DTDs 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 cannot 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

[8] Structured Documents, Document Transformations and Software Components.

application: we first obtain the DOM[9] [15] 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* [1]. 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. Fig. 4 sketches the process.

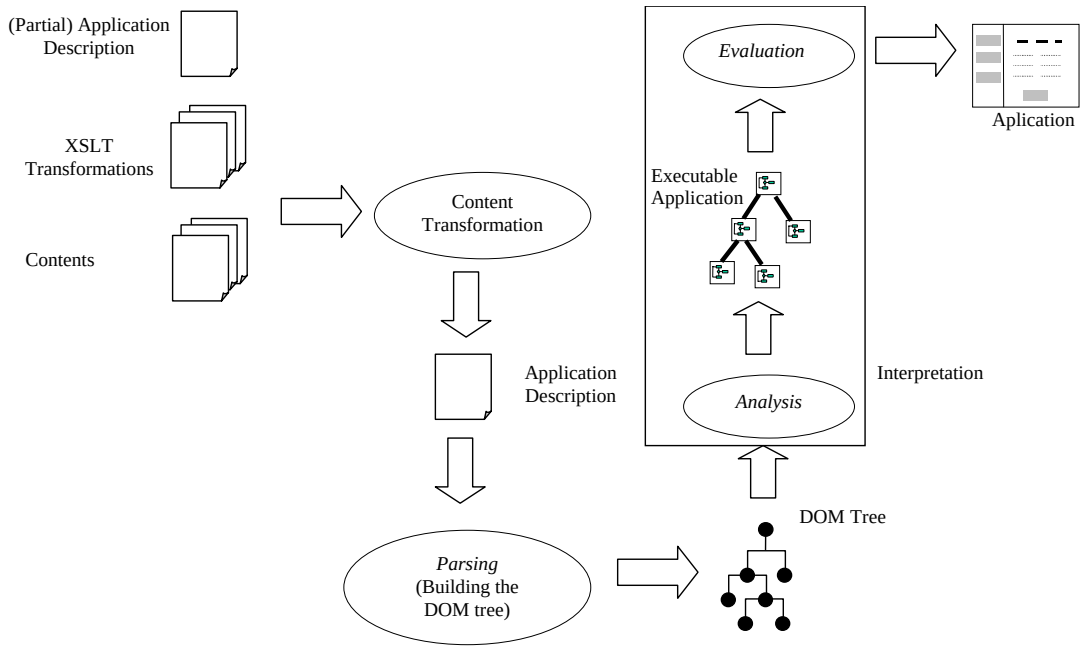

Fig. 4. Generation of a DTC application according to an *analysis and evaluation* strategy.

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 the techniques of construction of modular interpreters, like that described in [9].

At the moment we are working in order 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. Fig. 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 later used for the graphic presentation of the network) is described using the content DSLs. The DSL of the application 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 transforming the contents documents using XSLT transformations.

[9] Document Object Model.

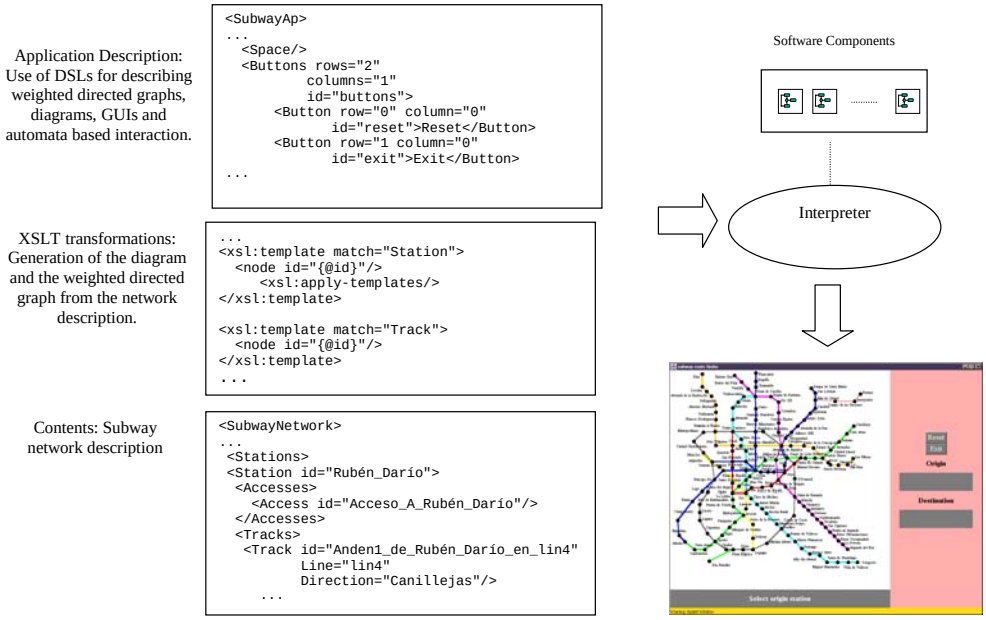

Fig. 5. Subway route finder application generated using DTC

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 an educational hypermedia, are based in 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 an 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 compose an application enables changes in one of them without affecting (or affecting slightly) to 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 documents that describe the whole application. The utilization of operational techniques like DTC facilitates the construction of those interpreter by means of the combination of simpler ones.

At the moment we are studying how to extend APG in order to support all the expressive power of *Pipe*. We also are analyzing the possibility of combining these techniques with other models and formalisms of information representation (as the relational model) and their effective incorporation to the declarative representation of the application. The following step is to deepen in the capacities 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

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

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

[3] Fernández-Manjón, B., Fernández-Valmayor A., Navarro, A. *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.

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

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

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

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

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

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

[10] Navarro, A., Fernández-Manjón, B., Fernández-Valmayor, A., Sierra, J.L. *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.

[11] Navarro, A., Fernández-Manjón, B., Fernández-Valmayor, A., Sierra, J.L. *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 abril, 2002. Published as Lecture Notes in Computer Science (Vol. 2306), Springer-Verlag, Berlin, 2002.

[12] Navarro, A., Fernández-Valmayor A, Fernández-Manjón, B., Sierra, J.L. *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.

[13] Sierra, J.L, Fernández-Manjón, B., Fernández-Valmayor, A, Navarro, A. *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.

[14] Sierra, J.L, Fernández-Valmayor, A., Fernández-Manjón, B., Navarro, A. *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.

[15] World Wide Web Consortium. *Document Object Model (DOM) Level 2* (several volumes). http://www.w3c.org/DOM/DOMTR, 2000.

[16] World Wide Web Consortium. *Extensible Markup Language (XML) 1.0 (Second Edition)*. http://www.w3.org/TR/REC-xml,2000.

[17] World Wide Web Consortium. *Extensible Stylesheet Language (XSL) Version 1.0*. http://www.w3.org/TR/xsl/,2001.

[18] World Wide Web Consortium. *Synchronized Multimedia Integration Language (SMIL 2.0)*. http://www.w3.org/TR/smil20/, 2001.

[19] World Wide Web Consortium. *XML Linking Language (XLink) Version 1.0*. http://www.w3.org/TR/xlink/,2001.

[20] World Wide Web Consortium. *XML Path Language (XPath) Version 1.0*. http://www.w3c.org/TR/xpath,1999.

[21] World Wide Web Consortium. *XML Pointer Language (XPointer) Version 1.0*. http://www.w3c.org/TR/xptr/,2001.

[22] World Wide Web Consortium. *XML Schema Parts 0,1,2*. http://www.w3c.org/XML/Schema,2001.

[23] World Wide Web Consortium. *XSL Transformations (XSLT) Version 1.0*. http://www.w3.org/TR/xslt,1999.