XML BASED MOBILE SERVICES

Outi Marttila and Petri Vuorimaa

Telecommunication Software and Multimedia Laboratory,
Helsinki University of Technology,
P.O Box 5400, FI-02015 HUT, Finland.
omarttil@tcm.hut.fi and Petri.Vuorimaa@hut.fi

ABSTRACT

The most remarkable trends in communication have been the huge popularity of Internet and the growth of digital cellular telephony usage. There is a strong demand to combine these two in the form of mobile Internet access. This paper discusses the service implementation issues for the wireless environment. The requirements placed on the services and service development by the mobility are presented, and the usage of the next generation, XML based modeling languages in the wireless services is analyzed. The results are based on the experiments gained from the implementation of three demonstration services.

Keywords: XML, XSL, SMIL, DOM, ECMAScript, mobile multimedia.

1. INTRODUCTION

The GO-project - which is part of the MediaPoli concept at Helsinki University of Technology - is focusing on the issues of mobile Internet access and establishing a wireless service architecture. The "Wireless multimedia services" subproject in Laboratory of Telecommunications Software and Multimedia aims at studying the implementation issues of the mobile multimedia software architecture. The main reasons for the need of a new kind of software architecture are

- continuous wide band media,
- new communication protocols,
- abstract service modeling, and
- memory, processing speed, power consumption, and physical space constraints.

This paper concentrates on the service modeling by studying the usage of the next generation modeling languages - XML, XSL, DOM, ECMAScript, and SMIL - in the development of mobile services. Three demo services were implemented with the objective to find out how these languages can be utilized, and what kind of advantages, or on the other hand disadvantages, their use brings compared to the former techniques. The evaluation was done mainly from the viewpoint of the service developer. With a powerful modeling language the developer is able to describe all the information and functionality needed in the service. Also, an efficient language allows the reuse of documents and flexible control over the rendition of information. Besides, multimedia applications require that the modeling language is able to define both synchronization and interactivity.

Mobility brings new possibilities to the services, but places at the same time additional requirements on the service development. Because of the fundamental limitations on power, available spectrum, and mobility, wireless networks have less bandwidth, more latency, less connection stability, and less predictable availability [WAPForum99]. The wireless devices usually have restricted power consumption, less powerful CPUs, less memory, smaller and lower quality displays, and different input devices.

In consequence of the mobility, the following factors have to be taken into account when designing the services to the wireless environment:

- size, resolution, and colors of the screens
  The size, resolution, and colors of the user interface are limited by the features of the display.
- data input and navigation

Input is done with a stylus and/or small amount of buttons instead of a mouse and a keyboard.

- size of the document files
  The size of the documents cannot increase, largely because of the constrained network and the limited memory and CPU capacity of the device.
- limited processing on the client device
  The memory and CPU place restrictions on how much of the processing of the document can be done on the client device.

The features of the display and user input facilities affect mostly the designing of the user interface. The size and resolution restrict the amount and size of elements that can be placed in one screen. Consequently, the placing of the components of the user interface has to be planned carefully. The formatting language should support this by allowing a flexible method for determining regions with size and position where the components can be placed. If the navigation is done with buttons, the environment for the service development must, by means of scripts or the application software, support it by handling key strokes and providing a way to show the location of the cursor.

Two other factors, the size of the documents and their processing, have to be considered besides the developer from the view of the whole architecture; how the actual information and styling related to it is described and handled. Besides the design decisions made by the developer - to avoid the size of the documents increasing too large - a technology used to describe the information should be able to do it flexibly and compactly.

2. TECHNICAL BACKGROUND

2.1 XML

To meet the new requirements of web publishing, W3C has developed the Extensible Markup Language (XML). XML was designed to deliver structured, possibly complex content over the web while still being easy to implement [Bosak97]. The goal was to develop a language that has the flexibility of SGML and the simplicity of HTML.

XML is primarily a meta-language for describing other markup languages. It neither specifies a fixed tag set nor the semantics of the tags, but allows users to define their own set of tags and the structural relationships between the tags. A document type definition (DTD) may be associated with the document, but it is not required. XML as such does not contain any functionality, but is used as a data description, interchange, and storage format.

Two basic properties of XML documents are well-formedness and validity [Bray98]. The well-formedness constraints control the proper syntax and structure of the document. Moreover, an XML document is valid if it is well-formed and it complies with the constraints expressed in an associated document type declaration.

XML linking consists of two markup languages, XLink and Xpointer. XLink is an XML language, which uses Uniform Resource Identifiers (URIs) to describe links between different files. XPointer complements XLink with the ability to address specific parts of elements or data in an XML document. Together, they allow multidirectional links, links to multiple resources, link databases, links that point specific places inside the documents, and links to and from read-only documents [Tauber99].

2.2 XSL

The Extensible Stylesheet Language (XSL) is the style language of XML. Since XML tags have no predefined semantics, XSL is used to describe how the elements are presented. XSL specification is divided in two separate parts [Deach99]. The actual XSL specification defines a vocabulary of formatting objects (XSL-FOs) that have the necessary base semantics. XSL Transformations (XSLT) specification defines a language for transforming the original XML document to the document that is composed of the elements having formatting semantics.

The conversion of XML to the presentation structure is done by XML/XSL processor. It takes an XML document, and constructs the source tree from the document. Using the XSL stylesheet, it constructs a separate tree, the result tree, which is composed of formatting elements. The result tree does not have to be composed of formatting objects introduced in XSL specification, but it can consist of any XML elements. However, the document resulting from the transformation must always be a well-formed XML document.

The basic building block of XSL transformations is the template rule, specified with `xsl:template` element describing how the original XML element node is converted into the element node that can be formatted, styled, and displayed [Clark99]. It consists of two parts, a matching part and a formatting part. The matching part identifies the XML node in the source document to be formatted, and the formatting part produces part of the result tree by applying formatting to the nodes.

An XSL formatting object represents a particular kind of formatting information, which is applied to the content of the formatting object. The formatting vocabulary is built on the basis of Cascading Style Sheets (CSS) and Document Style Semantics and Specification Language (DSSSL). Over 90 percent of the XSL formatting properties are defined in CSS. However, XSL extends CSS, e.g., to allow pagination and frame based structure. The extensions are done by adding new values to CSS properties, by splitting CSS properties into several new properties, or by creating completely new properties.

2.3 DOM and ECMAScript

Document Object Model (DOM) is a platform- and language-neutral interface that enables programs and scripts to dynamically access and update documents [Apparao98]. DOM Level 1 recommendation determines the objects for representing HTML and XML documents, and a basic interface for their manipulation. DOM Level 2, which under construction, builds on the DOM1 adding, e.g., interfaces for stylesheets, an event model, a query interface, and a DTD interface [Apparao99].

ECMAScript is a scripting language designed by standardizing organization ECMA on the basis of JavaScript. The goal of the language is to provide a standardized, unified definition of the languages, which are modeled after Netscape JavaScript [ECMA98].

2.4 SMIL

Synchronized Multimedia Integration Language (SMIL) is an XML based language that allows integrating a set of independent multimedia objects into a synchronized multimedia presentation [Hoschka98]. SMIL is able to describe the temporal behavior of the presentation, the layout of the presentation on a screen, as well as hyperlinks between media objects.

In a SMIL presentation, the media elements (e.g., images, audio clips, video clips, animations, and formatted text) are referenced from the SMIL document with Uniform Resource Locator (URLs) and are determined to be played in parallel or in sequence. The design goal of the language has been simplicity: it should be easy to create and edit presentations using a standard text editor.

The basic structure of a SMIL document is very similar to HTML document. A SMIL document is represented by a `smil` element which can contain `head` element describing the appearance and layout of a presentation and `body` element describing the timing and content information.

3. IMPLEMENTATION

3.1 Architecture

The stack in Fig. 1 represents the levels of the software platform of the wireless multimedia services from hardware at the bottom to the services on top. The most important application is the browser, which functions as a platform for the services. The services are modeled using XML or XML based languages like SMIL.

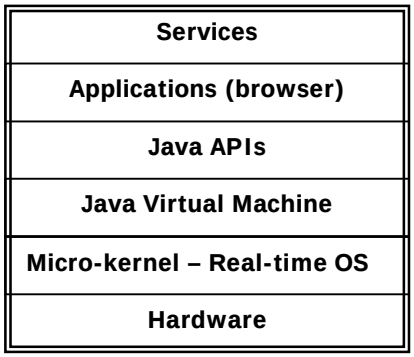

Fig. 1. Architecture.

Three demos of mobile services based on the architecture represented above were designed. The first two demos are exclusively implemented in terms of XML; the third one is modeled using SMIL.

Internet Explorer 5, supporting both XML and XSL as well as client-side scripts was chosen to be the client software for prototyping XML services. Since the browser is able to render HTML elements or elements styled with CSS, XML documents were transformed into DHTML with XSLT. User interaction was achieved besides hyperlinks with scripts using ECMA-compliant JScript by Microsoft as a scripting language.

3.2 Movie Demo

The first demo introduces a scenario of the service that movie theaters could offer to the mobile customers. The service provides

- the list of the films that are at the moment showing at the theaters,
- the schedule informing when and where the film is on,
- booking system for tickets,
- additional material related to the films, and
- descriptions of the theaters.

Navigation can be done with a stylus or arrow keys, and a link can be followed by pressing the “ok” key. The screen shots in Fig. 2-4 illustrate the main menu, the information of the movie, and the screen for buying the tickets.

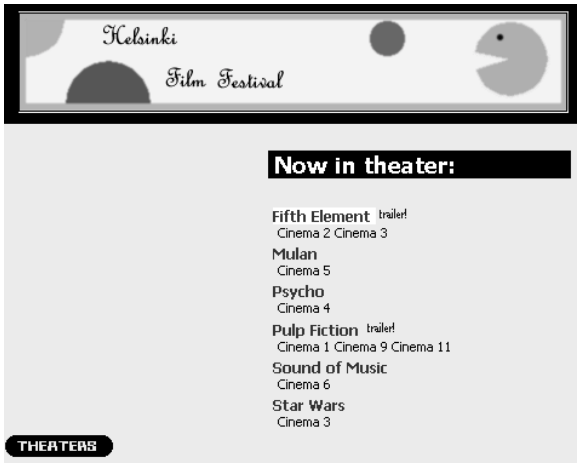

Fig. 2. The main menu of the movie demo.

Fig. 3. The screen presenting the information and showtimes of the film.

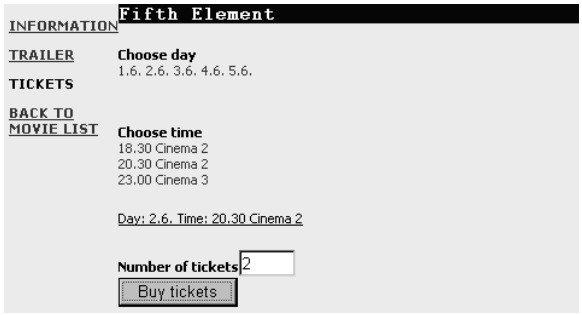

Fig. 4. The form for buying the tickets.

The data in this case is changeable and should be easily updatable and retrievable. XML provides a way to organize the pieces of information, so that the processing of information can be automated. XML is used to store the information, while XSL allows searching and selecting the fragments of information needed and presenting them in different ways. Data is stored structurally taking advantage of XML's ability to assign semantics to the information, and the desired information is picked from the document using XSL as a sort of query language. Using XSL and client-side scripts, it is possible to display the data in various forms without having to repeatedly request the server to send new pages.

The information content of the service consists of the movies and their schedules, theatre descriptions, and the booking situation of the shows. The movie data and theatre data are presented in separate XML documents. Fig. 5 presents the sample XML document describing the movie information. The information is contained in the `movies` element, which may contain zero or more `movie` elements, each of which represents a single movie and contains elements, which in turn include information related to the movie. Media elements like pictures and video clips are associated with the movie with the URL as in SMIL.

```
<movies>
     <movie name="Pulp Fiction">
          <trailer file="video1.asx"/>
          <showtimes>
            <theater name="Cinema 1">
               <day date="21.2.">
                         <time>18.00</time>
                         <time>20.30</time>
                    </day>
                    <day date="22.2.">
                         <time>18.30</time>
                         <time>20.30</time>
                    </day>
               </theater>
               <theater name="Cinema 9">
                    <day date="27.2.">
                         <time>16.00</time>
                    </day>
               </theater>
          </showtimes>
          <information>
            Tarantino's award-winning film….
          </information>
          <picture file="pulp.jpg"/>
     </movie>
     <!--  more movie items -->
</movies>
```

Fig. 5. XML example: the movie data.

Fig. 6 shows the basic structure of the stylesheet for the movie data. It is divided into five template rules. When the XML file is loaded for the first time, the template rule for the root element is processed. The rule constructs the HTML document with the `head` and `body` element. From the document body, it separates an area whose content changes as a result of

Fig. 7. The screen for attending the lecture.

user's actions. Inside the element determining the changing area, the `xsl:apply-templates` processing rule chooses the `movies` element for processing.

```
<xsl:stylesheet
xmlns:xsl="http://www.w3.org/TR/WD-xsl">
 <xsl:template match="/">
  <html>
    <head>
     <!--scripts and styles are included
here-->
    </head>
    <body>
     <div id="content">
      <!--area for the changing content-->
      <xsl:apply-templates select="movies" />
     </div>
    </body>
  </html>
 </xsl:template>
 <xsl:template match="movies">
  <!--goes through the movie elements in a
loop and forms the front page-->
 </xsl:template>
 <xsl:template match="movie">
  <!--forms the page representing movie
information-->
 </xsl:template>
 <xsl:template match="trailer">
  <!--forms the page containing the trailer-
->
 </xsl:template>
 <xsl:template match="showtimes">
  <!--forms the page for the ticket booking-
->
 </xsl:template>
</xsl:stylesheet>
```

Fig. 6. Structure of the stylesheet for the movie data.

Template rule for the movies element actually renders the first page, which includes the list of the movies, by selecting all `movie` elements with `xsl:for-each` element. Template rule for the root element is processed only when the document is loaded. Other rules are carried out as a result of the calls for `transformNode` method in scripts. This method applying to XML node object takes as a parameter an XML node, which represents an XSL stylesheet, and returns the transformed result nodes. In the scripts, the desired XML node is selected, and during the transformation resulting from the call for the `transformNode` method, the corresponding template rule is applied to the selected node.

3.3 Distance Education Demo

In distance education, students are not physically constrained to the same location as the instructor to receive instruction, but rather the instructional delivery is done by using audio, video, and computer technologies. The demo implemented here presents a distance education service, which provides lecture slides, video of the lecture, and a chat allowing communication between the students and the instructor. In teaching event, the instructor gives a lecture, which is filmed using a video camera. Together with the video there are slides of the lecture that can be read and scanned on the screen during the lecture. Chat enables students to interact in real-time. The screen for attending the lecture is presented in Fig. 7. By clicking the alternative views from the menu on the right side of the screen, it is possible to place only one of the elements (video, slides, or chat) on the screen.

The role of XML in the implementation is to describe the information in propriety document format so that it can be easily styled, accessed and processed. Though the number of documents may be rather large, a single stylesheet is used to display all the documents of the certain type. Most of the complexity - when coded in every document separately would result in considerable amount of redundant information - is included in the stylesheets.

The information of the service includes the courses that are available and lectures each course has. The information for each lecture includes slides and data related to the video and chat. To avoid the size of the documents growing too large, the information is split

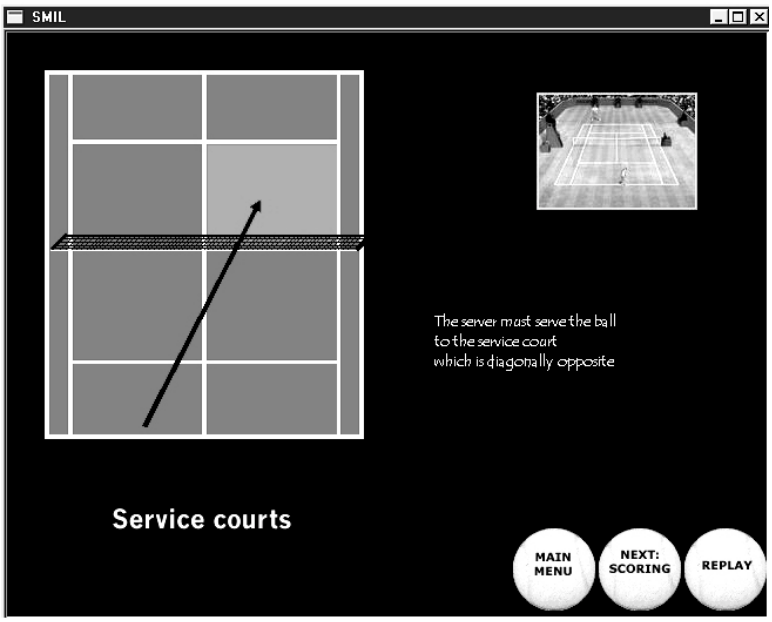

Fig. 9. Screen of the multimedia presentation.

into document containing the list of the courses, documents containing the list of the lectures for each course, and documents containing the data for a single lecture. The code sample in Fig. 8 presents the structure of the document containing the lectures of the course.

```
<lectures>
    <lecture>
     <subject>Opening of the course</subject>
     <date>6.1.1999</date>
     <url>lecture1.xml</url>
    </lecture>
    <!-- more lecture items -->
</lectures>
```

Fig. 8. XML description of the lectures of the course.

The structure of the stylesheets for the course list and lecture list is simple. With the `xsl:for-each` element all the `course` elements, or in the case of lecture list `lecture` elements, are picked and the link list is constructed using the content of the `url` elements.

The screen for attending the lecture is composed of two frames, the menu frame and the main frame where the actual teaching material is. Sources of the frames are static HTML documents and a specific stylesheet forms the HTML document containing these frames. Source of the menu frame contains scripts responding to the events. When it is loaded, a script opens the desired XML document containing the lecture data on the main frame. The URL of the XML document to be loaded is passed as a parameter to the script using a specific element in the course document. For video, chat, and slides there are separate stylesheets. The scripts in the menu document load the necessary stylesheet when user wants to change the view.

3.4 Multimedia Presentation Demo

The demo introduces an example of multimedia presentation with pictures, audio, and video. The theme of the presentation is tennis, aiming at introducing the game by giving the user information about the tournaments, players as well as the history and basics of tennis. GriNS player, which complies for the most part with SMIL specification and supports also various media formats, was used to test the presentation. Fig. 9 shows one of the screens of the demo.

The content is composed of still images, AVI and QuickTime videos, and WAV sounds. Media elements are joined together with SMIL taking advantage of its synchronization and linking capabilities. Code sample in Fig. 10 illustrates the SMIL implementation of one of the screens, which presents the history of tennis by showing texts with video. The implementation of other screens is basically very much alike, the complexity of the presentation depending mostly on whether complicated synchronization is needed.

4. CONLUSIONS

The implementation of the demo services showed that designing basic XML documents is very simple. However, when the information is more complex and supposed to be processed automatically, more attention should be paid to how the information is structured.

```
<smil>
 <head>
  <layout>
    <root-layout background-color="#000000"
width="650" height="500"/>
    <region id="text" width="407"
height="249" top="10" left="10"/>
    <region id="v1" width="136" height="196"
top="30" left="470"/>
    <region id="link" width="72" height="72"
top="420" left="575"/>
  </layout>
 </head>
 <body>
  <par>
    <a href="main_menu.smi"><img src=
"main.gif" dur="indefinite"
region="link"/></a>
    <audio src="history_sound.wav"/>
    <par>
     <seq>
      <img dur="4s" src="text1.gif"
region="text"/>
      <img dur="8s" src="text2.gif"
region="text"/>
      <img dur="8s" src="text3.gif"
region="text"/>
     </seq>
     <video src="history.avi" region="v1"/>
    </par>
  </par>
 </body>
</smil>
```

Fig. 10. Sample SMIL document.

4.1 XML Analysis

In the movie demo, all the information is in a single XML document and the screens of the service are generated only by varying the presentation of the information. For example, a HTML implementation, which requires that each of the screens is presented as a separate document, would have several documents. One problem with the XML implementation is that the file size can become too large, in which case the information should be split at the server, and serve the client only with a fraction of the document.

In the distance education demo, the main advantage of using XML is the reusability: the same document format is applied to number of documents. Moreover, the style information can be declared in a single stylesheet that is suitable for all documents. In this demo, the user is also able to modify the view by changing a stylesheet.

The most significant advantages of XML can be summarized in the following three elements:

- flexibility
  XML can be used for many different purposes in various services, systems, and platforms providing also an interchange format between applications.
- reusability
  Document format can remain unchanged among number of documents, while the content is changing.
- "intelligence" (the semantics of information)
  The processing of the information can be automated.

XML is, first of all, a device and platform independent method for describing information. For the wireless services that are accessed from the devices with varying features this is very important: the information coded in XML is always the same, but the presentation can be chosen according to the facilities of the device.

4.2 XSL Analysis

Though the functionality of XSLT can be achieved using DOM and scripts, the transformation language provides more simple and straightforward method to modify the structure of the document for presentation

As far as the support for interactivity and functionality is concerned, XSLT provides the `xsl:functions` element that allows declaring the functions that are interpreted during the transformation. For the scripts providing functionality after the transformation (e.g., to handle events) the formatting language must have the appropriate element where the scripts can be included. Especially, the interactivity would be increased with the ability to invoke XSL transformations programmatically.

XSL-FOs and CSS have very similar features and there is a lot of overlapping features among them. For example, both are able to determine the layout of document and the positions of separate components, which was found important for the mobile services. Since the implementation of both XSL-FOs and CSS in a browser would lead to rather heavy software, it is probably reasonable to implement only one of these in a lightweight browser.

The key questions from the view of wireless services are whether the formatting model of XSL is too complicated, and whether all the formatting objects introduced in the specification are needed. In addition, the basic structure of the style declarations of XSL-FOs and CSS is slightly different. XSL-FOs are elements that have predefined semantics and property attributes. CSS does not add new elements, but declarations are attached to the original elements by means of attributes. Because of this difference, XSL-FOs are probably more straightforward to use with XSLT whereas CSS is more useful for attaching style declarations directly to the original elements when transformation is not needed and the original document can be remained unchanged. When the document is not transformed, CSS is more simple to use, because use of XSLT results in large amount of extra code. Considering the restrictions of the wireless devices, the use of XSLT increases the size

of the style document and part of the processing of the document is transferred to the client device. However, the extra demands on the processing power cannot be considered very remarkable.

4.3 DOM and ECMAScript Analysis

Perhaps the most severe incompatibility problems at the moment on the web are resulting from the different JavaScript and document object model implementations. DOM and ECMAScript are the first efforts to standardize the scripting environment.

Especially DOM is vital for the service development determining how the structure of the documents is described and how the parts of the document are accessed. DOM1 specification provides a good basis for the basic manipulation of the documents, but the present DOM1 does not contain all the needed features until DOM2 with, e.g., event model and support for stylesheets is completed.

4.4 SMIL Analysis

Since its approval, SMIL has been adopted widely - considering the number of implementations which have been made by both non-commercial groups and commercial companies. Though it is possible to achieve SMIL's functionality with scripts that control the displaying and moving of the objects in a presentation, SMIL enables a wider group of developers without specific programming skills to make presentations by providing a very simple technique for describing how different elements relate over time.

The disadvantage of SMIL is that it is a separate multimedia document format and its features cannot be utilized in connection with other XML based languages. Moreover, the same functionality, but with more sophistication, can be achieved with other techniques. SMIL does not offer any new features or more powerful technology, but its strength is that it is a standard and independent of a specific vendor or software. Also, specific software is not required for developing SMIL presentations, but rather a text editor is adequate. However, with the present SMIL browsers the software-neutrality is not fully realized, because browsers support very varying set of features and a presentation designed with one browser seldom plays predictably with some other browser. Another disadvantage is that the flexibility of the SMIL presentations is actually very limited. All the interactivity is achieved with the simple hyperlinks and there is no opportunity to add scripts or constructs like frames that would allow updating only a part of the screen.

The developers of SMIL are aware of the weaknesses and working on it to improve the standard. The working draft of the next version of SMIL code-name "Boston" [Ayars99] has been recently announced. It improves the SMIL 1.0 by allowing the integration of timing into other XML based languages and including animation functionality, DOM for SMIL, and an event model.

REFERENCES

[Apparao98] Apparao V. et al., *Document Object Model (DOM) Level 1 Specification*, <URL: http://www.w3.org/TR/REC-DOM-Level-1/cover.html>, 1998.

[Apparao99] Apparao V. et al., *Document Object Model (DOM) Level 2 Specification*, <URL: http://www.w3.org/TR/WD-DOM-Level-2/>, 1999.

[Ayars99] Ayars J. et al., *Synchronized Multimedia Integration Language (SMIL) Boston Specification*, <URL: http://www.w3.org/1999/08/WD-smil-boston-19990803>, 1999.

[Bosak97] Bosak J., *XML, Java, and the Future of the Web*, World Wide Web Journal, Vol 2, No. 4, 1997.

[Bray98] Bray T. et al., *Extensible Markup Language (XML) 1.0*, <URL: http://www.w3.org/TR/1998/REC-xml-19980210>, 1998.

[Clark99] Clark J., *XSL Transformations (XSLT) Specification Version 1.0*, <URL: http://www.w3.org/TR/1999/WD-xslt-19990421>, 1999.

[Deach99] Deach S., *Extensible Stylesheet Language (XSL)*, <URL: http://www.w3.org/TR/WD-xsl>, 1999.

[ECMA98] ECMA, *ECMAScript Language Specification*, <URL: ftp://ftp.ecma.ch/ecma-st/e262-pdf.pdf>, 1998.

[Hoschka98] Hoschka P., *Synchronized Multimedia Integration Language (SMIL) 1.0 Specification*, <URL: http://www.w3.org/TR/REC-smil/>, 1998.

[Tauber99] Tauber J., *XML after 1.0: You Ain't Seen Nothin' Yet*, IEEE Internet Computing, Vol. 3, No. 3, May-June 1999.

[WAPForum99] WAP Forum, *The Wireless Application Protocol*, <URL:http://www.uplanet.com/pub/feb99WAPWP.pdf>, 1999.