# Enhanced Web Components and Connectors Description for Authoring e-Learning Environments

Karim Zeramdini, Yassin Rekik, Denis Gillet
Swiss Federal Institute of Technology, Lausanne (EPFL)
LA - School of Engineering, ME - EPFL - Ecublens, CH - 1015 Lausanne, Switzerland
EMAIL: {karim.zeramdini, yassin.rekik, denis.gillet}@epfl.ch

***Abstract*-This paper presents an approach that allows educators without programming skills to build and deploy Web-based experimentation environments. Conscious of the need to move existing Web Based Courses Management Tools (WBCMT) from document-centered framework to services-centered one we propose a Structured Composite Active Document (SCAD) model as a hybrid model to assemble heterogeneous and active components. Based on this SCAD model, educators can use the authoring tool to integrate and manage in a visual way on-line educational resources dedicated to sustain hands-on practice in engineering education. This approach is integrated in the development of the second version of the *eMersion* environment, whose first version is currently used at the School of Engineering, Swiss Federal Institute of Technology in Lausanne (EPFL).**

## I. Introduction

The impressive growth of the Internet over the last few years has provided educators with an effective medium to transmit knowledge via new learning approaches. Thanks to the availability of Internet, students can get to their training material at any time and any place. This flexibility lets students become more motivated to learn [1] and allows them to adapt the learning scenario to their skills and profiles.

Conscious of the added values brought by the new information and communication technologies, numerous educators have started to provide pedagogical content over the Internet as complement to the classical face-to-face classroom delivery.

To efficiently provide pedagogical content through the Web, educators need to define an appropriate delivery scenario and architecture. As non-IT specialists, most educators encounter difficulties in carrying out such a task. One of the consequences of these difficulties is that quite often the educators over simplify the learning content [2]. To overcome this problem [3], educators have access today to various Web-based Course Management Tools (WBCMT). WebCT [4], BlackBoard [5], TopClass [6] and IBM Learning Space [7] are examples of WBCMT that allow educators to pick from a rich set of pedagogical content and applications, and integrate them with just a few mouse clicks without programming effort. Hence educators do not need to spend time thinking about content representation, storage and publishing. They can focus on content quality and pedagogical strategies.

Despite all the advantages of the existing WBCMT, their major drawback is that they are restricted to the deployment of static multimedia components such as texts, images, and videos. The term ‘static’ is used here in contrast to *active components*, which can perform internal calculations and/or adaptations and can communicate with other components.

One example of context where classical WBCMT are not able to help educators in deploying pedagogical components is the deployment of on-line experiment resources to strengthen hands-on practice in engineering education. In this case, the pedagogical content relies on active components with internal calculation, communication possibilities (data import/export), and user interfaces. Examples of such components are Applets for the remote manipulation of physical devices or PHP interfaces to access simulation tools.

In the context of the *eMersion* project [8], carried out at the Swiss Federal Institute of Technology in Lausanne (EPFL), we are interested in developing and deploying on-line experimentation facilities. Hence, we have proposed a new service-centered WBCMT. The underlying approach relies on a generic model that supports Web-based experimentation resources. This model allows educators to assemble and deploy heterogeneous and active pedagogical components without programming effort.

The paper is organized as follows. Section II gives a short overview of the *eMersion* project with a special focus on the requirements for on-line experimentation deployment. Section III introduces the architecture defined for the composition and the deployment of active pedagogical components. It also introduces the new structured active document model. Section IV describes the authoring and deployment environment, which is under development to implement the defined architecture. Section V concludes the paper.

## II. The eMersion Project

The *eMersion* project aims at providing a Web-based environment that supports hands-on experimentation through remote manipulation of physical laboratory devices and/or computer simulation tools. In the context of this project, we have developed a service-oriented WBCMT allowing definition and the deployment of a Web-based experimentation environment that we call the *Cockpit*. Thanks to this WBCMT, several cockpit environments have already been deployed and are currently used and well integrated in the framework of three engineering courses: Automatic control, Fluid mechanics, and Biomechanics.

A cockpit environment typically involves active components such as Java Applets for the remote and real-time manipulation of physical devices, PHP interfaces for the remote access to simulation tools and JSP interfaces for analysis and reporting. All these components are inter-connected so that they can exchange automatically data such as files, experimental results, and configurations.

The problem we have been facing is that cockpit environments were based on fixed pedagogical scenarios. In fact, after being generated and deployed, a cockpit environment could not be modified and/or extended to fit a new learning situation or user profile without deep maintenance actions. As a result, when using our first version of WBCMT, and even with some customization facilities supplied (interface language, windows positions, etc.), educators had to follow pre-defined templates and scenarios.

Our work is now concentrated on the development of a second version of WBCMT which allows educators to define, create and adapt on their own the experimentation environments they need by assembling heterogeneous and active Web components, without dealing with technical or implementation aspects.

## III. STRUCTURED COMPOSITE ACTIVE DOCUMENT MODEL: SCAD MODEL

Thanks to the integration of business components (Java applets, Flash applications, CGI Scripts, etc.) that can perform internal calculation and adaptation, documents become active. According to [9], “documents are active in the sense that they include or are linked to executable code and data sets designed to produce dynamic renderings; they are network-aware, in the sense that they have components that are updated (automatically or manually) via remote, Web-based facilities”. A composite document can be seen as a collection of heterogeneous pedagogical components that can also be distributed. Therefore, within the composite active document class, pedagogical digital components can be composed, ordered, and correlated to support communication and training. In addition and to greatly facilitate their exchange and reuse [10][11] we have chosen composite active documents to be structured documents in the spirit of XML.

Before introducing the composition and deployment model, we need to define clearly the concept of active Web components. Indeed, these components can be considered as a combination of three different aspects: content, service and user interface. Combining these three aspects is the major contribution of our model. In fact, a lot of work has been carried out to tackle the problem of Web components assembly. However, no integrated solution has been proposed. One of the three aspects has always been given greater importance according to the underlying vision as described below.

- The document engineering vision: in this case, Web components are regarded as multimedia objects such as images, videos, and animations. Assembling such multimedia components has been the objective of many research projects. The majority of the contributions were based on the Allen relationship [12] with different extensions, such as [13], [14] and [15].

- The software engineering vision: in this case, the focus has been put on application components with a special interest in assembling distributed code to compose distributed applications. Here, the service aspect is the most important contribution and research has concentrated on Component-based frameworks such COM+ [16], Enterprise Java Bean (EJB) [17], CORBA Component Model (CCM) [18] Web Services (WS) [19] and Web Services for Interactive Application (WSIA) [20].

- The HCI engineering vision: in this case, Web components are regarded as multimodal end-users interfaces such as browser windows in Web-based environment, buttons and forms. Assembling HCI components models in HCI engineering is based on the same assembling approaches as presented previously in the document engineering vision.

In order to integrate the three previously mentioned aspects, we have defined in the *eMersion* project a hybrid model allowing Web component definition, assembling and deployment. This model is called the SCAD model, where SCAD stands for Structured Composite Active Documents. This model is based on the following assumptions:

- Components must be self-descriptive black-box entities that encapsulate services, which are accessible only via well-defined interfaces.

- Components must provide one or several interfaces to be used and require one or several interfaces to communicate with other components.

- Components must be configurable in order to be used in different contexts.

- Components must be accessible through the Web (interfaces based on HTTP, SMTP, etc.).

- Components must be language and platform independent. So components developed with Java language and running as example on Mac OS must be able to communicate with another one developed using C# and running on a PC.

The SCAD model is based on five basic concepts: content, container, behavior, communication, and layer.

- The *content concept* is associated with on-line resources that educators need to build and deploy. Nowadays, many educators use Macromedia or Director to create content, and FLASH or SVG [21] for delivery of animations and multimedia content. So far, we have defined a class of on-line content (Web components), which were introduced in the SCAD model. These are: images, videos, audios, structured documents (HTML, XML, SVG, SMIL), Applets and Flash applications. To integrate components into a SCAD document, educators could use either a simple text editor or the proposed authoring tool. In both cases, the SCAD document created must conform to the SCAD model grammar. If the SCAD document is generated through the proposed authoring tool, educators do not have to verify its conformity with the SCAD model. The proposed authoring tool automatically handles such a task. Nevertheless, when using a simple text editor to create a SCAD document there is a need to validate its content with an XML parser.

- The *container concept* represents a grouping area in which one should integrate Web components. The container concept could be combined with the behavior and the layer concepts, as it will be detailed below. For instance, a container may be a browser window in a computer Web-based environment.

- The *behavior concept* allows educators to express rules whose actions are applied to containers and contents. As in UIML language [22], the *Behavior* elements rely on rule-based language. Each rule contains conditions and a sequence of actions. Whenever a condition is true, the associated actions are executed. The behavior concept aims at giving educators a rich way to express interaction between components, between components and containers, and between containers as illustrated in figure 1.

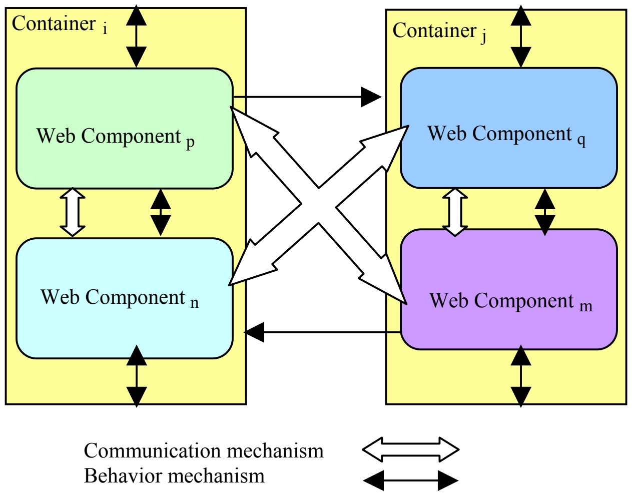

Figure 1: Behavior and communication mechanisms within SCAD model

So far, we have introduced into the SCAD model a set of rules and actions. The rules are: *OnMouseClick*, *OnMouseOver*, *OnResize*, *OnMove* and *OnClose*. The actions are: Close, *Open*, *Move* and *Refresh*. An action could also change attribute and element values. For example (see figure 2), when an appropriate action is triggered the *isVisible* attribute of a container element is changed to true: thus the container will be visible on the screen.

- The *communication concept* allows application components to share and exchange data (see figure 2). In fact, components, as small binary objects or programs performing specific functions, can operate with other components and applications by appropriate communication mechanisms. To establish connections between two components or more, each component must be a self-descriptive black-box entity that encapsulates services, which are accessible only via well-defined interfaces. Hence, to communicate, an input interface must be matched to an output one. The communication concept has not yet been completely introduced into the SCAD model and constitutes our area of special interest in future work.

- The *layer concept* facilitates the graphical authoring of the experimentation environment. Indeed, the layer concept is used to resolve container overlap problems during the authoring process. When a container is hidden (even partially) by another one it is difficult to edit its content. Hence, the layer concept permits to adequately manipulate the containers.

To define the SCAD model with its related concepts, we have adopted an XML-based scheme [23]. Figure 2 shows a simple example of a composite active document based on the SCAD model.

In this example, the container element includes seven attributes. The *id* attribute represents a unique container identifier. The *LocationX* and the *LocationY* attributes describe the container spatial position expressed in percent. The *Height* and *Width* attributes define the container size expressed in percent. When the *Resizable* attribute is set to *YES*, it is so possible to resize it in the deployed experimentation environment. The *isVisible* attribute allows one to choose whether the container is to be visible at the experimentation environment startup or not. The first container integrates an image. The second one a StructuredDoc element, which constitutes a kind of Web component that may be an HTML, XML, SMIL or SVG document. The *StructuredDoc* encapsulated elements are respectively:

- *File* element that specifies the type of document and its URL.
- *Grammar* and *Style* elements optionally incorporating the related document grammar and style files.
- *ZipImage* element optionally including the related document images.

The *Event* tag nested in the *Behavior* one specifies that when the image is clicked the *isVisible* attribute of the second container becomes true. More concretely, when the image is clicked the second container is shown in the experimentation environment.

```
<?xml version="1.0" encoding="UTF-8"?>
<SCAD xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
xsi:noNamespaceSchemaLocation="DACS.xsd">
<Title>Experimentation environment example</Title>
<CreationDate>2004-04-15</CreationDate>
<CreationTime>12:23</CreationTime>
<!-------------------------- CONTAINER CONCEPT-------------->
<Container id="Container1" LocationX="10%" LocationY="20%"
Resizable="YES" Visible="YES" HEIGHT="35%" WIDTH="35%">
<!-------------------------- CONTENT CONCEPT ---------------->
<Image id="Img1" SRC="image.jpg" LocationX="10%"
LocationY="10%"/>
<!-------------------------- BEHAVIOR CONCEPT --------------->
<Behavior id="behavior1">
<Rule id="rule1">
<Condition>
<Event type="OnClick" component-id="Image1"/>
</Condition>
<Action>
<Content Container-id="toggleable_container">
<NewProperty name="isVisible" value="true"/>
</Content>
</Action>
</Rule>
</Behavior>
<!-------------------------- LAYER CONCEPT --------------------->
<OnLayers>
<Layer number="0"/>
</OnLayers>
</Container>
<!-------------------------- CONTAINER CONCEPT-------------->
<Container id="toggleable_container" LocationX="50%"
LocationY="20%" Resizable="YES" isVisible="false" HEIGHT="40%"
WIDTH="40%">
<!-------------------------- CONTENT CONCEPT ---------------->
<StructredDoc Language="FR">
<File Type="HTML" URL="MyDocument.html" size="20"/>
<GrammarURL="grammar.xsd"/>
<Style URL="style.css"/>
<ZipImage URL="archive.zip"/>
</StructredDoc>
<!-------------------------- LAYER CONCEPT --------------------->
<OnLayers>
<Layer number="0"/>
</OnLayers>
</Container>
</SCAD>
```

Figure 2: SCAD XML fragment

## IV. AUTHORING AND DEPLOYMENTARCHITECTURE

Based on the SCAD model, we have started the implementation of a second version for our service-oriented WBCMT. In this version, the experimentation environment is defined in a descriptive way using a SCAD document. A SCAD-based document can be generated through the use of a "WYSIWYG" authoring tool and, then, used as input for the rendering tool. This architecture is presented in figure 3.

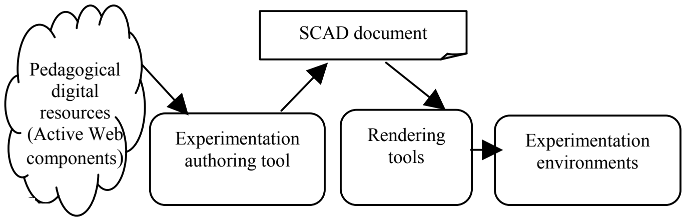
Pedagogical digital resources (Active Web components)
SCAD document
Experimentation authoring tool
Rendering tools
Experimentation environments

Figure 3: Authoring and deployment architecture

The authoring tool allows educators to intuitively assemble on-line resources. The SCAD document generated serves as input for one or several rendering tools in order to produce one or several experimentation environments. For instance, a first rendering tool is used to deploy a Web-based experimentation environment dedicated to desktop computer clients. A second serves to deploy Web-based experimentation environment dedicated to PDA clients. The use of a SCAD document as an intermediate and neutral format permits such a deployment flexibility. The authoring and the rendering tools are described in more detail below.

### *THE AUTHORING TOOL*

The authoring tool allows educators to integrate containers and Web components in a more visual and intuitive way. This means that an educator can drag a component from the palette (see figure 4) and drop it into the container. Then, in the container, s/he can modify the component properties using the properties form. The GUI gives a look and feel that is close to what they want to obtain when deploying the experimentation environment (see figure 5).

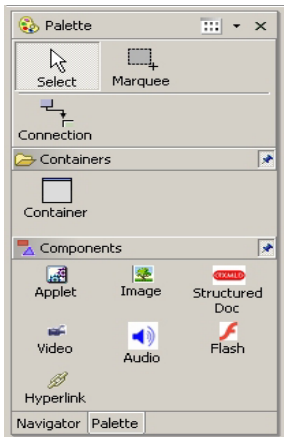

Figure 4: The authoring palette tool

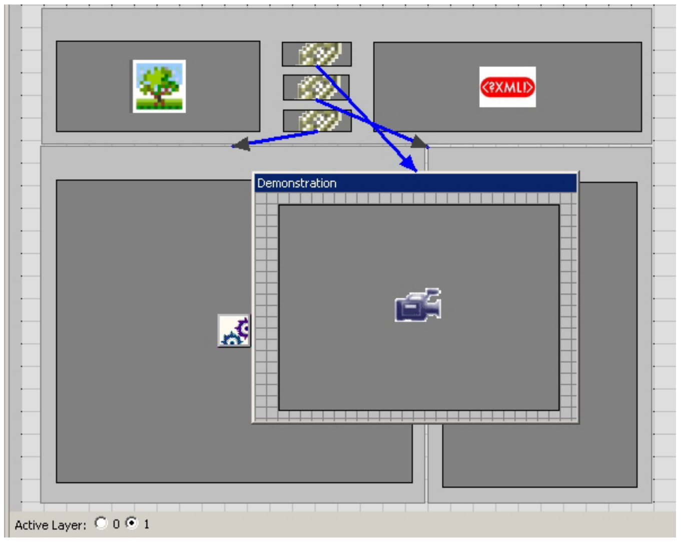

Figure 5: The experimentation authoring tool

According to the SCAD model, Web components must be the child elements of a container, i.e. educators first have to create at least one container in the workspace. They could realize the operation just by clicking on the container icon from the palette view and without releasing the right mouse button they create the container graphical interface. The same interaction approach could be performed for integrating a Web component within a container. To add Web component properties or container ones conforming to the SCAD model, the educator has to double click on it. A form containing the properties is shown. He has to fill in at least the required fields.

The authoring tool also allows educators to establish behavior relationships between Web components and containers. An educator has to click on the connection icon from the palette view and draw an oriented line from the Web component/container source of the interaction to the destination one. Then, while double clicking on the oriented line, s/he can customize the behavior source parameters and the behavior destination ones.

The layer graphical metaphor offers to educators a friendly means of using the authoring tools to create containers and to integrate Web components into them. It permits one to manipulate in different views containers and Web components that have to share the same spatial area when the experimentation environment is deployed. When the authoring tool detects spatial overlap between two containers, the last one created is put in a different layer. To edit the content of each container, educators have just to switch from one layer to another. In addition, educators can draw behavior-oriented lines between two containers, between containers and Web components, or between two Web components that are located in different layers.

To ensure the utility of the SCAD XML documents generated and their conformity with the SCAD model, the authoring tool manages two kinds of constraints: structured constraints and graphical ones. Structured constraints management handles elements nesting as well as element and attribute types. For example, if an element is defined as date type, the authoring tool generates an error when the educator inserts character strings. Meanwhile, graphical constraints management handles graphical arrangements, which have to be in accordance with the structured ones. Therefore, the authoring tool does not allow the educator to drag Web components, which must be dragged from the palette outside the containers.

### *THE RENDERING TOOL*

The experimentation-authoring tool allows the educator to generate quickly and easily SCAD documents conforming with the SCAD model. This document constitutes the first step toward the deployment of experimentation environments. The second step is to convert the high-level object-based description of the SCAD document into a Graphical User Interface (GUI) for display. One or several rendering tools can be used for this purpose depending on the educator needs. In other words, based on one SCAD document, many GUI experimentation environments can be rendered. So far, we have developed a first rendering tool that involves a DHTML popup to display the containers; JSP and JavaScript ensure interactivity between containers and contents.

The rendering tool is integrated as an embedded toolkit within the experimentation-authoring tool. Therefore, at the completion of the SCAD document, the educators can directly deploy the experimentation environment.

## V. CONCLUSIONS AND FUTURE WORK

This paper presents an innovative approach for authoring and deploying service-oriented Web-based experimentation environments in engineering education. This approach is based on a hybrid model, namely the SCAD model that considers experimentation environment as a composition of active Web-components. This model integrates classical multimedia components such as marked-up texts, images, and video, as well as active components with internal calculation and communication capabilities such as PHP interfaces, Java Applets, and HTML forms.

The SCAD model allows a descriptive definition of an experimentation environment, thanks to its five major concepts: content, container, behavior, communication, and layer. This model has been formalized using an XML schema. Thus, an experimentation environment, in our approach, must be defined as an XML document conforming to the XML-based SCAD schema.

Based on this model, we have developed an authoring and deployment environment allowing novices to create graphically and intuitively new experimentation environments. This authoring and deployment environment is basically composed of a graphical authoring tool that generate SCAD documents and a rendering tool that uses SCAD documents as an input to generate experimentation environments.

In the current version of our authoring and deploying environment, we have only considered a limited set of Web components. We have also defined only basic communication mechanisms. Currently, we are working on extending our model and our environment with new components types and new communication facilities.

Finally, we are also working to improve the graphical interface for authoring SCAD documents and generating experimentation environments. In fact, due to the fact that we are targeting novice users, we have to translate all the SCAD concepts and mechanisms into comprehensive and intuitive graphical interfaces that can be exploited by such users. The challenge was, and still, to offer an intuitive authoring interface including adapted metaphors and interactions that allow novice users to benefit from the SCAD model advantages and make technical and syntactic aspects transparent for them.

## VI. Acknowledgement

The *eMersion* project was funded by the Board of the Swiss Federal Institutes of Technology in the framework of its New Learning Technologies program. The authors would also like to thank A.V. Nguyen for fruitful discussions.

## VII. References

[1] B. C. Chiu, Y.T. Yu, “Promoting the Use of Information Technology in Education via Lightweight Authoring Tools”. Proceedings of the International Conference on Computer in Education (ICCE’02), December 03-06, 2002. Auckland, New Zealand.

[2] R. Oliver, J. Herrington, “Online learning design for dummies: professional development strategies for beginning online designers”. In P. Barker & S. Rebelsky (Eds.), Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications, ED-MEDIA 2002. Norfolk, VA: AACE, (pp 1500-1505)

[3] D. N. Perkins “What constructivism demands of the learner”. Educational Technology, 31(8), 19-21, Volume XXXI, Issue 9 September 1991.

[4] WebCT, http://www.webct.com/ last visited: April 19th 2004

[5] BlackBoard http://www.blackboard.com/ last visited: April 19th 2004

[6] TopClass, WBT System, http://www.wbtsystems.com/ last visited: April 19th 2004

[7] LearningSpace, http://www.lotus.com/products/learnspace.nsf last visited: April 19th 2004

[8] eMersion project: visit http://emersion.epfl.ch/

[9] S. W. Alman, T. Christinger “Active Documents: the New Generation of Digital Documents and the Implications for E-Learning”. Proceedings of The Eighth Sloan-C International Conference on Online Learning, November 8, 9, & 10, 2002. Orlando, Florida, USA.

[10] V. Quint, I. Vatton “Making structured documents active”. Electronic Publishing - Organization, Dissemination and Design, 7(2): 55-74. 1994.

[11] V. Quint, I. Vatton, J. Paoli “Active Structured Documents as User Interfaces”. Vol. User Interfaces for Symbolic Computations. Springer Verlag, 1995.

[12] J. F. Allen. “Maintaining Knowledge about Temporal Intervals”. Comm. ACM, 26(11): 832-843, November 1983.

[13] P. King, H. Cameron, H. Bowman, and S. Thompson. “Synchronization in Multimedia Documents”. In Jacques Andre, editor, Electronic Publishing, Artistic Imaging, and Digital Typography, Lecture Notes in Computer Science, volume 1375, pages 355-369. Springer-Verlag, May, 1998.

[14] F. Vernier, L. Nigay “A framework for the Combination and Characterization of Output Modalities”. Proceedings of the 7th International Workshop on Design, Specification and Verification of Interactive Systems (DSV-IS), June 5-6, 2000, Limerick, Ireland.

[15] SMIL 2.0 Synchronized Multimedia Integration Language, visit http://www.w3.org/TR/smil20/ last visited: April 19th 2004

[16] COM+: http://www.microsoft.com/com/tech/COMPlus.asp last visited: April 19th 2004

[17] EJB: Enterprise JavaBeans, visit http://java.sun.com/products/ejb/ last visited: April 19th 2004

[18] CCM: CORBA Component Model, visit http://www.omg.org/ technology/documents/formal/components.htm last visited: April 19th 2004

[19] Web Services, Visit http://www.w3.org/2002/ws/ last visited: April 19th 2004

[20] WSIA : Web services for interactive application, approved as an OASIS Standard August 2003. http://www.oasis-open.org/committees/download .php/3343/oasis-200304-wsrp-specification-1.0.pdf last visited April 19th 2004

[21] SVG, Scalable Vector Graphics, Visit http://www.w3.org/TR/SVG/ last visited: April 19th 2004

[22] UIML: User Interface Markup Language, visit http://www.uiml.org/ last visited: April 19th 2004

[23] XML schema visit http://www.w3.org/XML/Schema