ADAPTIVE INFORMATION SYSTEMS FOR THE WEB 2.0: DEVELOPING A MOBILE LEARNING ARCHITECTURE

DAVID PARSONS*, SEBASTIAN SCHROEDER
INSTITUTE OF INFORMATION AND MATHEMATICAL SCIENCES
MASSEY UNIVERSITY AUCKLAND, NEW ZEALAND
d.p.parsons@massey.ac.nz

ABSTRACT

The Web 2.0 is a collection of concepts and patterns that attempt to define the nature of contemporary web applications. It has much in common with Service Oriented Architecture (SOA) concepts, which rely heavily on the use of XML and interoperable schemas, as well as embracing open web based APIs. One of the key design patterns of the Web 2.0 is 'software above the level of a single device', whilst one of its key concepts is the data driven nature of services and the core value of specialised databases. In both cases, XML has an essential role to play, both for data and metadata. In this paper, we explore some aspects of developing information systems in the context of the Web 2.0, focusing particularly on device adaptivity, data transformation and interoperability with other services, including the mashup approach. We describe a generic architecture for adaptive web applications and apply this to the development of a prototype mobile learning system.

KEYWORDS

Mobile Learning, Adaptive Web Applications, Web 2.0, XML, Java, Service Oriented Architecture.

INTRODUCTION

The Web 2.0 is a term that has become widely used since the first Web 2.0 Conference in 2004. Although it might be categorised as an umbrella marketing term rather than a specific technology or architecture, some authors (notably Tim O'Reilly (O'Reilly, 2005)) have given it some concrete specifications through a set of published principles, practices and patterns. Many publications that discuss the Web 2.0 focus on rich user interfaces, in particular the use of AJAX (Asynchronous JavaScript and XML). However, there is a much broader set of technologies and architectural issues to consider.

The key ideas underlying the Web 2.0 are the web as a software platform, user and developer communities and service oriented architectures. Some authors use the term *Programmable Web* to refer to similar concepts, but stressing the Application Programming Interfaces (APIs) that are used in 'mashups'. A mashup, in web terms, is an application that uses various APIs to mix together web based data from different sources. The term mashup is also used to refer to music ('bastard pop') created by combining elements of previous recordings, and web mashups have the same philosophy of attempting to create a new artefact that is more than the sum of its parts. Some mashups, for example Virtual Places (Kothari, 2005), use a number of different APIs to produce an overall application. APIs used on the site include Amazon, Alexa, FeedMap, Flickr, MapPoint, MSNSearch and VirtualEarth. Other mashup web applications restrict themselves to using a single API, but reorganise the available data services to suit their requirements.

In the Web 2.0 a participatory architecture is important. In such systems, contributors build their own shared databases, for example in file sharing programs or web logs (blogs) that maintain persistent associations with other web resources via permalinks, which are unique, persistent, Uniform Resource Locators (URLs). Interactivity is enhanced by the use of (various implementations of) RSS (an acronym that has two roots, Really Simple Syndication and Rich Site Summary) and podcasts, a term generally used to refer to audio files delivered via RSS, but which may in fact provide content in a range of different media types such as video or text files. Underlying these architectures is the concept of specialised databases being the core item of value in a system, with the opportunity to reuse these databases in new combinations of services.

One key Web 2.0 pattern is *software above the level of a single device*, both in terms of the server (as a syndicating end point for mashup services) and the client (many different types of device for a given service) with those clients able to contribute to the database that underlies services. In this paper we explore some of the key themes in the Web 2.0 and demonstrate some of these themes through a prototype mobile learning system, building on a generic description of an adaptive web application architecture. In this architecture we emphasise the role of the eXtensible Markup Language (XML) in programming above the level of a single device, and leverage a number of APIs that can assist in building adaptive web applications, and consuming and providing web services.

KEY THEMES IN THE WEB 2.0

O'Reilly lists seven principles of the Web 2.0, along with eight patterns and seven 'core competencies of Web 2.0 companies' (O'Reilly, 2005). In addition, MacManus and Porter (2005) provide seven trends in Web 2.0 application design. Since a number of these overlap, and they have been described elsewhere, in this paper we will provide an overview of the key themes, which we summarise as: application development as a web based, community activity, data driven web applications and adaptivity of the client and the server.

APPLICATION DEVELOPMENT AS A WEB-BASED, COMMUNITY ACTIVITY

Traditional software construction is based on teams of developers working through a planned project lifecycle to provide software that is intended to meet a well defined set of requirements. These applications are generally intended primarily for 'in house' use, though they may expose certain features to customers. For example, software used by banks is primarily used internally, but may expose web based services enabling customers to perform certain transactions on their accounts. The software development model in the Web 2.0 is much more loosely coupled, based on cooperation rather than control. Applications are inherently web based, so that the web itself becomes the development platform. Software is in a 'perpetual beta' state, continuously evolving while being tested in the field by its users. Application content is not generated centrally but syndicated from multiple sources, using APIs that positively encourage 'hackability' and 'remixability'. Programming tools must therefore be lightweight and encourage assembly from reusable components and services. The community of users not only tests software but contributes to it, either directly (by developing server side applications) or indirectly (because their actions, such as blogging or contributing to Wikis, contribute to the content of the application).

DATA DRIVEN WEB APPLICATIONS

The first generation of the web emphasised presentation, but the Web 2.0 puts more emphasis on content, using the term 'Data as the Next Intel Inside' to indicate the importance of data as being the measure of value. To enable the interoperability of data between different contexts and applications, the eXtensible Markup Language (XML) is increasingly being used, both as a way of representing content within an application and between applications (via web services). Since XML provides for self-describing, semi structured data, it enables the development, over time, of metadata to more usefully describe and link together content from across the web. XML based web services underlie most of the available APIs for accessing services in mashup web applications, where the API enables the developer to build web applications that gather data from XML messaging and assemble applications from multiple XML based sources. The role of XML in the Web 2.0 is essentially to provide services that cannot be supported using more traditional approaches to web applications that provide content directly in the Hyper Text Markup Language (HTML). There are four broad categories where XML provides such services (Bosak, 1997):

1. Applications that require the Web client to mediate between two or more heterogeneous databases.

2. Applications that attempt to distribute a significant proportion of the processing load from the Web server to the Web client.

3. Applications that require the Web client to present different views of the same data to different users.

4. Applications in which intelligent Web agents attempt to tailor information discovery to the needs of individual users.

All four of these categories are relevant to the Web 2.0, underlying the importance of managing content in XML format in the development of contemporary web applications.

ADAPTIVITY OF THE CLIENT AND THE SERVER

There are three aspects to adaptive web applications; content, navigation and presentation (Koch & Rossi, 2002). Adaptive content means providing different content for different users. For a client of a web application, adaptation is based on user profiles, where the user's role in adaptation is largely passive and the adaptation takes place on the server. However we can also consider adaptive content from the perspective of the client of a service, or a set of services. In this case there is active programming on the part of the user, assembling their own web applications from other services using publicly available APIs. Adaptive navigation may also be based on the user profile and involves changing the selection, presentation and/or ordering of the anchors that link web based components together. Adaptive presentation means changing the presentation depending on device, among other things. XML is important here as it provides a presentation neutral way of representing the underlying content of an application. Equally important are the tools for transformation, such as eXtensible Stylesheet Language Transformations (XSLT) and the eXtensible Hypertext Markup Language (XHTML), a well formed and validatable form of HTML, which is a common target of XML transforms. XHTML is designed for consumption by browsers on various types of device, including, increasingly, mobile devices.

ADAPTIVE PRESENTATION ON MOBILE DEVICES

Enabling a system to work across range of devices means having an awareness of the likely boundaries of device capability and size used by the mobile learner. Weiss (2002) identifies three generic categories of screen size; small mobile phone windows, medium smart phone or pager sized screens and large Personal Digital Assistants (PDAs). Of course in practice we must regard these three size categories as being sets of samples from a continuous range, with many different screen dimensions available. In addition to screen size, resolution also varies widely, from the minimum benchmark of 96 by 54 pixels in the Java Mobile Information Device Profile (MIDP) specification (JSR118, 2002) to about 240 by 320 pixels upwards for PDAs. In addition to these concerns, user may have control over certain display settings.

DEVICE CAPABILITY AND MARKUP

Coupled with the form factor of devices, we also have to consider the technical capability of the device. Different devices will offer different types of support for technologies such as the Java 2 Micro Edition (J2ME), Windows Mobile or browser plugins such as Flash. In addition there are a number of different types of mark-up for mobile devices. These include the WML (Wireless Markup Language) used with Wireless Access protocol (WAP) phones, which comes in two major versions, WML 1.x and WML 2.x, developed from the earlier Handheld Device Markup Language (HDML). Mostly in Japan, but also in other markets, there is cHTML (Compact HTML) for iMODE phones, and increasingly the XHTML compliant subsets XHTML-Basic and XHTML-MP (Mobile Profile). Figure 1 (From Golding (Golding, 2004)) shows the various types of markup and their genealogy

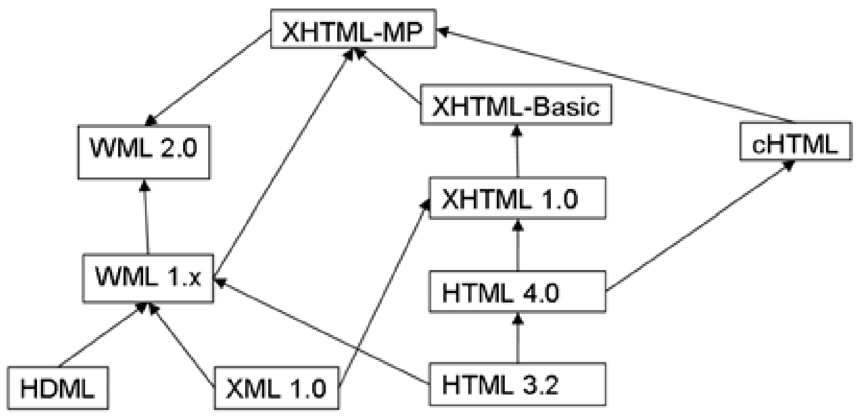

Figure 1: The genealogy of markup languages.

Given all these complexities, we have to devise a strategy that is able to deliver the same (or similar) content to a wide range of devices. In order to facilitate this we need to apply the Web 2.0 concept of programming above the level of the device. However at some point the device type needs to be identified so that a suitable transformation of back end data can be performed for that device. Multiple page formats need to be generated on the server for different types of client, either via the 'User-agent' information sent within the micro-browser's Hyper Text Transfer Protocol (HTTP) request header or more sophisticated Custom Configuration / Personal Profile (CC/PP) data. Once the device type has been identified, service resources represented in raw form in XML can be transformed into markup using eXtensible Stylesheet Language Transformations (XSLT) and/or Cascading Style Sheets (CSS).

The number of variables in device presentation, including screen size, resolution, user controlled settings and markup languages, is so large that a hand coded approach to adaptivity of presentation on mobile devices is not realistic. Therefore we need to look at reusable components operating above the level of the device that can provide these services. Fortunately, a number of technologies exist that encapsulate presentation adaptivity using server page tag libraries. These enable most XML processes to be coded using a few relatively simple tags.

ADAPTIVE WEB APPLICATION ARCHITECTURE

In the previous sections we introduced some key features of the Web 2.0 along with some technologies that can assist us in the development of adaptive web applications. In this section we describe an architecture for adaptive web applications that leverages XML as a data representation format within an adaptive architecture and integrates the service oriented aspects of the Web 2.0. Perhaps the most important aspect of such an architecture is the introduction of a middleware layer that integrates disparate data sinks and sources into a cohesive XML management process. This enables a system to manage requests and responses in a consistent way, regardless of whether the client request is for a web service or presentation markup, and whether our data sources are local databases or remote web services. An outline of the architecture and its core processes is shown in Figure 2.

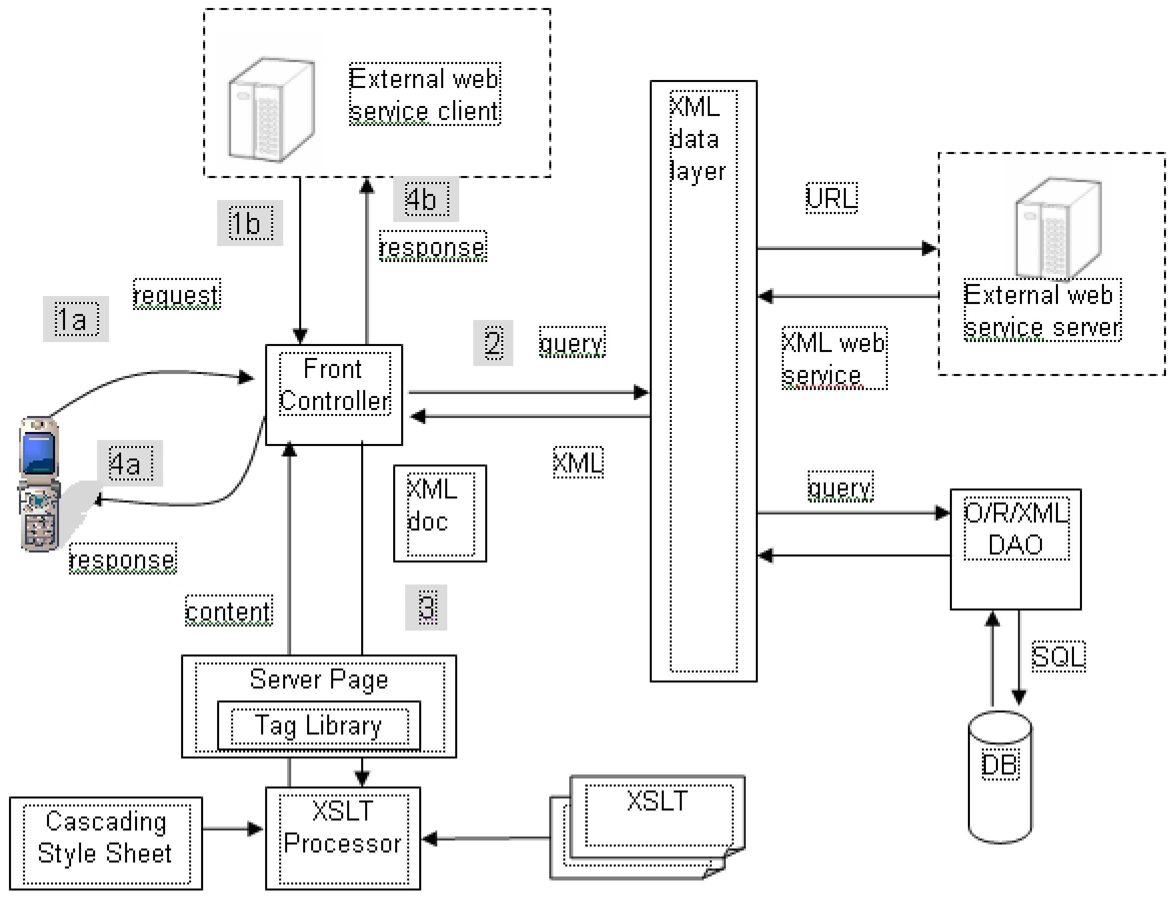

Figure 2: An architecture for adaptive web applications.

In this architecture, client requests to a front controller component may come either from presentation oriented devices (1a) or external web service clients (1b). Depending on the requirements of the request, queries delegated via an XML data layer may either retrieve XML content from one or more external services running on other web servers, generate XML content from a local database via a suitable queries or combine multiple data sources (2). Where content is based on external web services, content will already be in XML format. Where content is held locally in a database that is not natively XML a layer of XML Object/Relational Data Access Objects (DAOs) is required used to perform the translation from relational data to object model to XML. The object model layer is required to preserve the integrity of the underlying entities in the system. This enables multiple XML documents to be generated from a model that is created from a single database query. The XML content returned from the XML data layer is transformed via an XSLT processor utilizing a specified XSL transformation stylesheet (3). This transformation should be performed via a server page using tag libraries to ensure encapsulation and reuse. Once the content is transformed it can be transferred to the device in an appropriate mark-up format for processing by the client. This may be a device oriented markup (4a) or a web service (4b). Of course many mobile devices can both process markup and act as end points for web services.

This type of XML centric architecture has a number of benefits. First, we can gain reuse of both external web services and middleware components. Second, we can provide adaptivity that integrates both client device mark-up and web service endpoints. From a business perspective it provides maximum leverage of external services and APIs while gaining maximum potential distribution across all possible client types. Encapsulation of data access and transformation should provide benefits in terms of maintainability and cost. For example, separating out concerns between XML, object models and the database, rather than directly generating XML from the database, can assist in the reuse of legacy data sources and provide added security through additional layering.

A MOBILE LEARNING EXAMPLE

In this section we describe an executable architecture developed for a mobile learning system based on the architecture described in Figure 2, using Java tools for the implementation. We might summarise the main requirements of a mobile learning platform as enabling learner communities to evolve by supporting the multi way exchange of content and being usable across a range of different devices. From our previous discussion we can see that these have emerged as core themes of the Web 2.0.

LEARNER COMMUNITIES AND MULTI WAY EXCHANGE OF CONTENT

To engage the learner and draw them into a virtual community, two way interaction is an important feature of a mobile learning environment. On the one hand there will be communication from the system to the learner. This may involve profiled notifications sent that are important to the learner's context (Koschembahr, 2005). Other types of communication may be messages from fellow mobile learners or role play participants, general multicasts or more personalised communications from a learning facilitator (Rogers et al., 2004). On the other hand, there will be communication from the learner, which may involve submitting test answers, contributing to discussion groups, sending messages, writing blogs or adding to Wikis or sharing other information as part of pooled resources. Therefore mobile learning systems must provide the ability to upload resources to a central repository in a number of different formats. These resources may include rich media elements such as photographs or videos that can contribute to the learner's portfolio (Deviney & Koschembahr, 2004) as well as being used as generally shared content (Trifonova, Knapp, Ronchetti, & Gamper, 2004).

The learner's ownership of content is a motivating factor. They should be able to create, review and edit the content while the social aspect that is central to learning in a mobile learning community can support the shared corporate culture. In addition, the organisation benefits from the growth of answer gardens developed from pooled resources (Ackerman & Malone, 1990).

With these requirements in mind, a mobile learning architecture needs to provide some technical mechanism for content sharing and interactivity using a common data format. This can be achieved in large part by the use of mashups and reusing existing functionality based on XML technology. Depending on the area of learning, there may be a number of possible services that could be integrated. The key architectural choice is using a framework that enables the integration of such mashup services with custom content using common tools and representations, regardless of the specific data requirements of client devices and the direction of data transfer.

USE ACROSS A RANGE OF DIFFERENT DEVICES

We have already discussed some of the technical platforms that might be used in a mobile learning system. In addition, we should be aware of some issues specific to mobile learning that relate to device type. First we have to be aware of the lowest common denominator of our learners' devices. In some contexts, (e.g. professional training) the learners may all use the same device, specified by the employer, for example some of the case studies described by Gayeski (2002). Otherwise we might specify a minimum platform that we are prepared to support, for example the Mobile Learning Engine (Meisenberger, 2004) requires a WAP/Java platform as the minimum specification. Another important issue in mobile learning is the provision of different ways of achieving the same learning, so that we might need, for example, to provide the same content as a text file and an audio file, to cater both for different personal preferences and different learning contexts.

SYSTEM ARCHITECTURE

In addition to the general concerns of the architecture already described, there are some specific issues encountered in our Java-based mobile learning system. One issue is that there are few currently available external web services that could be used for mobile learning, though there are one or two, for example Wikipedia. Another issue is that where a service is provided via an API (as opposed to a low level XML web service) that API will be language specific and therefore must be compatible with the mobile learning server's software platform in order to be used. In our prototype, which uses the JBoss Java application server, any leveraged APIs must be written in Java. One example of this is Flickr (2006), which includes a Java API (among others) and enables a number of interactive applications to be developed based around blogging and picture sharing.

The O/R/XML Data Access Objects (DAOs) in the current prototype are JavaBean based, enabling them to be used via tag libraries in JavaServer Pages (JSPs). These beans provide an object model to organise the requested set of data. This is particularly useful where a client page will include data from multiple tables that have some kind of associative relationship. In cases where the content relates to subject navigation, the beans form a composite pattern (Gamma, Helm, Johnson, & Vlissides, 1995) with the composite objects organising submenus and the leaf objects leading to learning content URLs that may, for example, be a page that downloads a Java program, or contains mashup services. XML transformations are performed via JavaServer Pages (JSPs) using tags from the JSP Standard Tag Library (JSTL).

Initial routing of the HyperText Transfer Protocol (HTTP) request from the client is done via a front controller JSP that parses request parameters and generates an appropriate query to the XML data layer. The eventual response depends on the device, which can be identified by the HTTP request. There are a number of Java based technologies that can assist us in adapting server-side content to the client device, but the one we have used in our prototype system is the Wireless Abstraction Library (WALL), a JSP tag library that builds on the Wireless Universal Resource File (WURFL) (Passani & Trasatti, 2002-6). WURFL is able to recognise user agent information and identify different devices, while WALL generates device specific markup. Figure 3 shows the basic process of transformation required to convert content stored in a database into a page adapted to the client device. First, a data access object (DAO) is used to process the client's HTTP request by executing a query against the database and returning an XML document of the requested content. This XML document is transformed using an XSL processor by applying a stylesheet that adds the necessary WALL tags to the resulting document. This document is the content of a JSP, which is then processed by the application server's JSP engine. During this process, the WALL tags will be processed and turned into the appropriate markup for the client device.

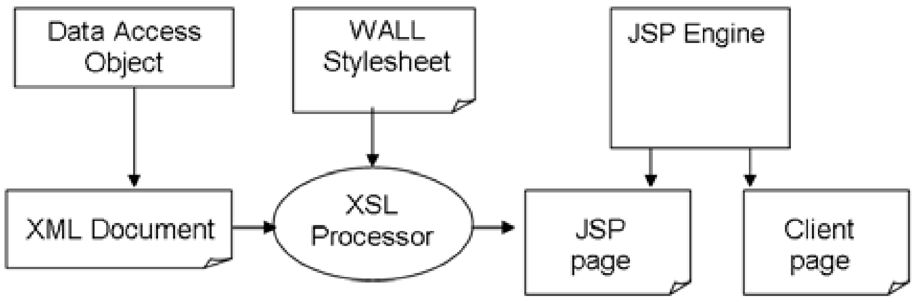

Figure 3: The process that transforms database content into adaptive presentation using the WALL tag library.

By using the WALL device aware library we are able to provide a single XSLT transform for many different types of device. However we might in addition choose to provide our own customised transform for particular types of device. For example, if there was a requirement to provide mobile learning content as desktop e-learning, then a transform could be used that would take advantage of the rich client technologies available on the desktop. Many Web 2.0 applications leverage AJAX and Flash, though there are may other possibilities. At the least, we might apply a Cascading Style Sheet (CSS) to manage the presentation in the browser. Where the client is a service end point, and the system provides content as a web service, the XSLT transform would not be generating presentation mark-up but XML documents suitable for web service use. In this case the provided mark-up should utilise standard schemas for learning content such as SCORM (ADL, 2004).

CONCLUSIONS, RELATED AND FURTHER WORK

In this paper we have described an adaptive web application architecture that supports the transformation of local or service based content into client-adaptive output formats. This architecture takes into account both current thinking on the technologies of web based applications and the encapsulation and reuse of standard libraries. In order to explore the practicalities of this architecture we have implemented a prototype mobile learning system that is able to generate intermediate XML content from learning content stored in a relational database and transform that content using tag libraries. We have also integrated some elements of Flickr as an example mashup service that meets some of the interactivity requirements of a mobile learning system.

Other authors have described similar systems that utilise adaptive architectures to deliver mobile learning. Sharma & Kitchen (2004) describe the role of web services in a mobile learning architecture, which has much in common with the services aspect of the architecture described here. However they focus specifically on the development of a mobile learning framework from a standard web services perspective rather than the role of mashups in a wider architecture. Goh et al (2003) describe an adaptive mobile learning architecture based on XML transformation, which has much in common with our prototype. It does not, however, integrate a relational database, and rather than using a fully adaptive library takes the approach of targeting three specific types of client device. Capuano et al (2004) take advantage of the device adaptivity support provided with the Microsoft platform, and augment this with the use of with SMS as an interactivity mechanism. Although this example uses different technology to our prototype, it has much in common with its conceptual approach. Syvänen et al (2005) describe a similarly adaptive system. Of interest here is their inclusion of contextual information for the learning content, an aspect that we have not included in our own prototype. Holzinger et al (2005) use the Mobile Learning Engine to develop a system that leverages J2ME. Again there is some similarity to parts of our own system, where J2ME applications can be launched to suitable devices.

We have described an architecture that is based on Java, XML and server side tag libraries. Others have used similar architectures or the Microsoft platform. An alternative approach is to use an XML framework such as Cocoon. This Java-based tool transforms XML, utilizing sitemaps that define by extensions (e.g. *.wml) which XSL templates are used. Libraries similar to WALL can be used in conjunction with Cocoon, which would be particularly useful in the context of using XML databases such as Xindice, for which direct support is provided

The current system provides an executable architecture (i.e. an architectural prototype) that demonstrates the key technical features of an adaptive mobile learning system. The next step in its development requires construction of learning content in quantities appropriate for usability testing. The system will then be deployed on a public server that will enable testing via the Internet using any data enabled mobile device.

REFERENCES

Ackerman, M., & Malone, T. (1990). Answer Garden: a tool for growing organisational memory. *Proceedings of Conference on Office Information Systems, Cambridge, Mass.*, 31-39

ADL. (2004). *SCORM*. Retrieved January 26th, 2006, from http://www.adlnet.org/scorm/index.cfm

Bosak, J. (1997). *XML, Java, and the Future of the Web*. Retrieved February 7th, 2006, from http://www.ibiblio.org/pub/sun-info/standards/xml/why/xmlapps.htm

Capuano, N., Gaeta, M., Miranda, S., & Pappacena, L. (2004). A System for Adaptive Platform-Independent Mobile Learning. *Proceedings of 3rd Annual MLearn International Conference (MLEARN 2004), Rome, Italy*,

Deviney, N., & Koschembahr, C. V. (2004). *Learning Goes Mobile*. Retrieved Sept. 16, 2005, from http://www.workindex.com/editorial/train/trn0402-02.asp

Flickr. (2006). *Flickr API Documentation*. Retrieved February 9th, 2006, from http://www.flickr.com/services/api/

Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, Mass.: Addison-Wesley.

Gayeski, D. (2002). *Learning Unplugged - Using Mobile Technologies For Organizational Training and Performance Improvement*. New York: AMACOM.

Goh, T., Kinshuk, & Lin, T. (2003, December 2-5, 2003). Developing an adaptive mobile learning system. *Proceedings of International Conference on Computers in Education, Norfolk, VA, USA*, 1062-1065

Golding, P. (2004). *Next Generation Wireless Applications*. Chichester: Wiley.

Holzinger, A., Nischelwitzer, A., & Meisenberger, M. (2005, 8-12 March 2005). Mobile Phones as a Challenge for m-Learning: Examples for Mobile Interactive Learning Objects (MILOs). *Proceedings of Third IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOMW'05), Kauai Island, HI, USA*, 307-311

JSR118. (2002). *Mobile Information Device Profile for Java™ 2 Micro Edition, Version 2.0*. Retrieved Sept 14, 2005, from http://jcp.org/aboutJava/communityprocess/final/jsr118/index.html

Koch, N., & Rossi, G. (2002, 3-7 July). Patterns for Adaptive Web Applications. *Proceedings of 7th European Conference on Pattern Languages of Programs (Europlop 2002), Irsee, Germany*,

Koschembahr, C. (2005). *Optimizing Your Sales Workforce through Mobile Learning*. Retrieved Sept 14, 2005, from http://www.learningcircuits.org/2005/apr2005/vonKoschembahr.htm

Kothari, N. (2005). *Virtual Places*. Retrieved February 7th, 2006, from http://apps.nikhilk.net/VirtualPlaces/

MacManus, R., & Porter, J. (2005, May 4th 2005). *Web 2.0 for Designers*. Retrieved February 7th, 2006, from http://www.digital-web.com/articles/web_2_for_designers/

Meisenberger, M. (2004). *Mobile Learning Engine*. Retrieved February 13th, 2006, from http://drei.fh-joanneum.at/mle/start.php?sprache=en&id=0

O'Reilly, T. (2005, 30th September 2005). *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*. Retrieved February 7th, 2006

Passani, L., & Trasatti, A. (2002-6). *WURFL*. Retrieved February 9th, 2006, from http://wurfl.sourceforge.net/

Rogers, Y., Price, S., Fitzpatrick, G., Fleck, R., Harris, E., Smith, H., et al. (2004, June 1-3, 2004). Ambient Wood: Designing New Forms of Digital Augmentation for Learning Outdoors. *Proceedings of Third International Conference for Interaction Design and Children (IDC 2004), College Park, Maryland, USA*,

Sharma, S., & Kitchens, F. (2004). Web Services Architecture for M-Learning. *Electronic Journal on e-Learning, 2*(1).

Syvänen, A., Beale, R., Sharples, M., Ahonen, M., & Lonsdale, a. P. (2005, 28-30 November 2005). Supporting Pervasive Learning Environments: Adaptability and Context Awareness in Mobile Learning. *Proceedings of IEEE International Workshop on Wireless and Mobile Technologies in Education (WMTE 2005), Tokushima, Japan*,

Trifonova, A., Knapp, J., Ronchetti, M., & Gamper, J. (2004). Mobile ELDIT: Transition from an e-Learning to an m-Learning System. *Proceedings of World Conference on Educational Multimedia, Hypermedia and Telecommunications*,

Weiss, S. (2002). *Handheld Usability*. Chichester: Wiley.