

Adaptive Educational Hypermedia Content Creation: A Web Service based Architecture

Maram Meccawy¹, Craig Stewart² and Helen Ashman¹

¹School of Computer Science and Information Technology, University of Nottingham, Jubilee
Campus, Wollaton Road, Nottingham, NG8 1BB, UK
{mzm,hla}@cs.nott.ac.uk

²Department of Electronic Engineering, Queen Mary, University of London
Mile End Road, London, E1 4NS, UK
craig.stewart@elec.qmul.ac.uk

Abstract. The penetration of Adaptive Educational Hypermedia (AEH) within the educational community has been very poor so far. In part this is due to the complexity of the authoring process. But it also influenced by the often short ‘lifetime’ of the AEH software itself. We see AEH system interoperability as a route to address these problems. This paper builds on previous limited interoperability work, between the AEH systems MOT and WHURLE, by using open web standards. We describe a Web Service using Web Services Description Language and SOAP that is capable of converting MOT materials to WHURLE. Moreover this Web Service is flexible and easily extensible, and we hope that this will mark the first step in developing an interoperable middleware for AEH conversions.

1 Introduction

Adaptive Educational Hypermedia (AEH) [3] aims to deliver personalised and appropriate learning materials to each learner. These materials can be adapted to a multitude of variables, such as a learner’s background knowledge, personal preferences, learning styles or grade scores (to name but a few). There are many distinct AEH systems, many of which have been developed and used in a research environment, such as AHA! [8], MOT [7], TANGOW [4], and WHURLE [10]. Each of these systems adapts to a slightly different ‘view’ of the learner, e.g. WHURLE stores each learner’s knowledge level (if they are considered a beginner, intermediate or advanced learner) at the granularity of each domain. Compare this to MOT’s more flexible approach that includes recording the knowledge level for each domain concept. This is a fundamental difference just between two of the current AEH systems in use. Multiply this by the number of AEH systems and add in that there will be far more than one such difference between each system and you will see that the levels of complexity involved in working with AEH systems can be very great. Any teacher (assume a non-technical expert who wishes to use an AEH system) who is considering authoring an adaptive lesson faces many decisions, such as:

- Which AEH system should they use?
- What knowledge domain(s) does the lesson require?
- Does any previous material exist (and is it in a usable condition)?
- What are the objectives of the lesson and how are they to be achieved for a heterogeneous group of learners?
- Which traits of a learner are to be modelled?
- Will the learner's data be gathered implicitly (without the learner's knowledge) or explicitly (information is requested from the learner)?
- How many versions of the same material need to be created, e.g. if an AEH system produces an adapted lesson for two different groups of learners, does the teacher have to create two lessons for every one they intend to deliver?
- What are the rules for adaptation? Can the author of the lesson modify these in any way?
- How are the various versions to be presented to the learner, and does the learner have any control over this?

Hence authoring for an AEH is a complex and time-consuming process. An author may well select a specific AEH system to learn how to use and then spend a large amount of time creating materials for this system. But what then happens if the chosen AEH system stops being developed or maintained? The author can either continue to develop for a system that will slowly go out of date and will one day cease to function, or they can move to a new system that is likely to be incompatible (in authoring methodologies and learning material structure) with the previous one.

Recently there has been work in generating a new authoring paradigm, one that moves away from the 'create once, use once' approach described above [12]. Only interoperability between AEH systems will address these authoring issues; only when an author can develop learning materials in one system and then have them delivered in another will AEH begin to enter the mainstream of education.

This paper suggests using a Web Service based approach to achieving this interoperability. An initial implementation between two AEH systems, MOT and WHURLE, is described that builds upon previous research into conversion between these systems [13]. The aim is to use this first step to learn more about the issues involved in creating a web service that can convert the from one system to another. This service can then offer to convert AEH data that matches certain criteria (as specified by the web service) into another system. This would be the first step in creating a middleware system that can transfer adaptive learning materials transparently between systems.

2 MOT & WHURLE

2.1 MOT

MOT [7] is a generic web-based AEH system based on the LAOS framework [6] for authoring of adaptive hypermedia, and implements a simplified version of the LAOS layers. Using MOT, authors can create domain concept maps (DM), containing the actual learning resources clustered as content alternatives, and also create the lessons themselves (Goal & Constraint Maps, GM), based on these domain maps, that allow a restructuring and filtering of the learning materials. These contents are stored in a MySQL database.

2.2 WHURLE

WHURLE [10] is a discipline independent, XML based AEH system. WHURLE uses atomic constructs, known as *chunks*, to store the learning content. These chunks are organized by a *lesson plan*, each lesson plan create a default pathway through the content. This default pathway is filtered according to the adaptation rules and the learner's user profile. The result of this is a personalized *virtual document* that delivers the lesson most appropriate to the learner.

2.3 MOT to WHURLE conversion

Enabling interoperability between two such very different systems is not straightforward. The first steps towards this were taken in [13], but this approach was limited. Our research enabled a direct conversion of learning materials using MOT as the authoring system and WHURLE as the delivery system. However the conversion program itself was an off-line, command-line based system. To transparently implement a 'create once, use often' paradigm for authoring, it would be much better to create a web-based system. The results of this research are presented in this paper.

3 Web Services: WSDL and SOAP

3.1 Web Services

In the last few years, web services have become more popular for application developers, especially in the business sector.

A *web service* is a software application, which is identified by a URL like ordinary websites, but can be accessed remotely by another application. The difference between web services and websites that makes web services unique is, in fact, the type of interaction that they can provide [9]. Thus, some argue that web services are the "*next evolution of the web*" [2]. Web services provide a solution to a major problem in the computer world: interoperability. Interoperability is provided by allowing different applications from different sources to communicate with each other without time-consuming customized coding. Since all communications are in XML, the services are not tied to any specific operating system or programming language. Therefore, C++ communicates with Perl, Java with PHP and Mac with Unix or Windows.

The power of web services resides in the fact that each web service implements a capability that is available to each other, or to other applications, via standards, networks and protocols.

The W3C¹[14] defines web services as "*a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.*" [15]

Another definition [2], states that a web service "*describes a standardized way of integrating web-based applications using the XML, SOAP, WSDL² and UDDI³ open standards over the Internet protocol backbone*". XML is used to tag the data, SOAP to transfer the data and exchanging information between computers, WSDL used for describing the services available and finally UDDI for listing what web services are available for a customer to use or buy. Web services transportation can be done over simple protocols (HTTP, SMTP, FTP, etc.); HTTP is currently the most commonly used web service protocol.

Using this approach the authors chose to implement a MOT to WHURLE conversion, in such a manner that the Web Service would be easily extensible. The open nature of the Web Service description would enable any system conversion engineer to add additional AEH system data input and output streams. This would mean, for a teacher creating adaptive material, he or she would not have to worry about keeping up with the growing complexity and

¹ World Wide Web Consortium

² WSDL: Web Service Description Language

³ UDDI: Universal Description, Discovery and Integration

different releases of authoring and delivering of learning software, as these issues can then be delegated to a different level, or even solved automatically in some cases.

3.2 WSDL:

The adaptive e-learning material interchange solution we propose is based on WSDL architecture. WSDL is an XML document that conforms to a specification. All the services metadata is contained somewhere in this file, structured in such a way that will make it easy to understand what the data means.

In addition to a WSDL file being human readable, all that a programmer has to do to generate the code necessary to connect physically to services is to use an XML parser to extract data into local variables. This automatic code generation is one of the outstanding features of web services [9].

A WSDL document [14] *"defines services as collections of network endpoints, or ports. In WSDL, the abstract definition of endpoints and messages is separated from their concrete network deployment or data format bindings. This allows the reuse of abstract definitions"*. A WSDL document uses the following elements in the definition of network services: **Types**, a container for data type definitions using some type system (such as XSD); **Message**, an abstract, typed definition of the data being communicated; **Operation**, an abstract description of an action supported by the service; **Port Type**, an abstract set of operations supported by one or more endpoints; **Binding**, a concrete protocol and data format specification for a particular port type; **Port**, a single endpoint defined as a combination of a binding and a network address; and finally a **Service** is a collection of related endpoints.

WSDL is a cornerstone in the web services architecture because it provides a common language to describe such services plus it provides a platform for integrating those services.

3.3 SOAP:

SOAP, which historically used to refer to, Simple Object Access Protocol, (although this was ceased in SOAP1.2 [1]), is the second important item in the web service architecture. SOAP is an XML-based protocol for exchanging information between computers. Its job is to encode messages in a common XML format so that message can be understood at each end (client and service). It is a high level of abstraction, so that any operating system and programming language combination can be used to create a SOAP-compliant program.

Web services use SOAP as a logical transport mechanism for moving messages between services described by WSDL interface [5].

SOAP can be defined as a *"specification for a ubiquitous XML-based distributed computing infrastructure"* [9]. It is a stream of characters that are carefully created so that the programs on both sides of the transmission can understand exactly what the other side is saying. Those characters are XML documents that are embedded in the transport's request and response messages.

A SOAP message is composed of three parts, two of which are mandatory and a third which is optional. The mandatory parts are: SOAP envelope <SOAP-ENV: envelope> and SOAP body <SOAP-ENV: body>; The optional part includes SOAP's header <SOAP-ENV: Header>. The syntax of the SOAP grammar allows for instructions to be added to the header with the actual method call or XML document to be added to the body.

4 MOT2WHURLE Web Service

4.1 Introduction:

In order to create a web service from scratch the following stages are required in this process, (from the perspective of a service developer): create core functionality of the service (system); create WSDL service description of that system; create a SOAP service wrapper; deploy service onto a server and register new service via UDDI.

Since we already had the conversion program, the first step in developing the service (creating the core functionality) was skipped, and our work started by creating the WSDL file. Only the work on the two core elements of Web Services has been completed: WSDL and few SOAP examples.

4.2 Defining the WSDL abstract layer:

In order to create the *WSDL semantic abstraction layer* that allows for AEH content conversion between the two separate systems (MOT and WHURLE), the commonalities between the two systems had to be identified and described. Here we only sketch the ones directly relevant to the WSDL layer.

To describe these commonalities in WSDL we first have to create the abstract section by defining the WSDL types <wsdl:types> for both MOT and WHURLE data types.

4.2.1 MOT Lesson conversion

The adaptive MOT lesson conversion into an adaptive WHURLE lesson plan was done by extracting the basic elements from the two systems and re-composing them, Figure 1 shows an extract of a MOT lesson.

```
<xsd: element name="MOTlesson">
  <xsd:complexType>
    <xsd:sequence>
      <xsd: element name="lessonID" type="xsd:int" />
      <xsd: element name="ToplessonID" type="xsd:int" />
    </xsd:sequence>
  </xsd:complexType>
</xsd: element name="MOTlesson">
```

```
<xsd: element name="Sublesson">...
```

Figure 1: A snapshot MOT lesson in WSDL

The composition of WHURLE lesson plans, containing chunks, is sketched in Figure 2.

```
<xsd: element name="WHURLELessonplan">
<xsd:complexType>
  <xsd:sequence>
    <xsd: element minOccurs="1" name="level">
      <xsd:complexType>
        <xsd:sequence>
          <xsd: element minOccurs="1" name="page">
            <xsd:complexType>
              <xsd:sequence>
                <xsd:element minOccurs="1" name="WHURLEchunk">
```

Figure 2: A snapshot WHURLE lesson plan in WSDL

4.2.2 Message, operations and PortTypes:

Next, the main messages that are exchanged/ transmitted during the conversion process are described. Figure 3 shows the messages that allow the user to select the lesson Id for a MOT lesson to be converted.

```
<wsdl:message name="printLM">
  <part name="MOTLessonId" type="xsd:int"/>
</wsdl:message>
<wsdl:message name="printLMRespons">
  <part name="returnIdWhichLM" type="xsd:int"/>
</wsdl:message>
```

Figure 3: WSDL Adaptive lesson Conversion messages

Following the message description was the portType description which included the operations (functions) that are supported within the conversion service. An example is given in Figure 4, which describes the above messages as part of a single operation. This operation identifies the messages that need to be sent, and the message that can be expected in return.

```
<wsdl:portType name="MOT2WHURLE_Conversion_PortType">
  <operation name="MLesson_WLessonplan_conversion">
    <input message="tns:printLM"/>
    <output message="tns:printLMRespons"/>
  </operation>
```

Figure 4: A snapshot for the "MLesson_WLessonplan_conversion" operation

By describing the systems' data types, the messages, as well as the available operations within the portType, the abstract description of the WSDL file was

completed. Next, the description of the concrete part which includes the WSDL binding, port and service is conducted. This part is responsible for connecting to the abstract part physical binding between a client and a service.

4.2.3 Binding, Port and Service:

Binding, as mentioned earlier in section 3, is about how will the message be transferred on the wire, and it has two main purposes: linking the abstract and concrete elements in WSDL, and as serving as a container for information such as protocol and address of web service. The WSDL file is completed by declaring the "port" and "service" elements, which identifies the service's location.

4.3 Creating SOAP wrapper examples:

In order to test the service and exchange messages/information with its server, the SOAP messages and response files were created. These are XML files that as described earlier in this paper (section 3.3) consist of three elements: Envelope, header and header (optional). Below is an example of a SOAP request in figure 7 for the "PrintLMresponse" operation:

```
<?xml version='1.0' encoding='UTF-8'?>
<SOAP-ENV:Envelope
  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
  xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
  xmlns:xsd="http://www.w3.org/1999/XMLSchema">
  <SOAP-ENV:Body>
    <ns1:printLM
      xmlns:ns1="http://cs.nott.ac.uk/~mzm/MOT2WHURLE/"
      SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <MOTlessonId xsi:type="xsd:int">12</MOTlessonId>
    </ns1:printLM>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Figure (7): A snapshot of a soap request (printLM)

As can be seen from the previous examples the user enters an Id number for a lesson in MOT (here it is '12') to be converted into a WHURLE LP and the response message will carry this number to the corresponding function in the conversion program (service), which will result into displaying the equivalent LP in WHURLE after the conversion process occurs.

5 Discussion

In this paper we have presented an extension of our work in creating a new authoring paradigm, 'create once, use often'. The move towards this from a 'create once, use once' approach to AEH system authoring is in an attempt to address the issue of authoring complexity. AEH systems have yet to break into the mainstream educational culture, and this may in part be due to the cost involved in authoring for these systems. In allowing AEH systems to exchange their learning materials we do not remove the initial cost of authoring, but we can ensure that authors only ever have to go through this process once. Interoperable AEH systems will not only future-proof access to these materials, but will allow for a more flexible delivery of them – a student would no longer be forced to use the AEH system that their teacher chose, but could use one that they were already familiar with. This increases the choice for the learner and increases the potential for personalisation.

The work discussed here builds upon a previous off-line conversion system [13]. Whilst this previous work allowed the teacher to author in the more flexible and user-friendly MOT authoring system, and then have it presented in the WHURLE delivery system, this was not an open system. To extend the conversion to other AEH systems would require the creation of an entirely separate conversion program.

In applying a Web Service based architecture we address two problems of the initial conversion system. The first is that the conversion system will be available across the web – hence any author using MOT (which is also a web based system) can then access the service to convert their materials for use in WHURLE. Thus the conversion program no longer needs to be installed on every author's computer. Also, in using open web based standards, we open up the conversion system. Other AEH systems can input to the conversion web service using the definitions given in the service definition. Also other systems can be added to the 'output'. Indeed an area of future work that we will be following up is extending the service so that learning materials from MOT may be converted for use in SCORM compliant systems (such as Blackboard).

Another area that has yet to be addressed is that of multiple authoring systems. The goal of a 'create once, use often' paradigm, is that any AEH system can create materials and have them converted into any other system. Our current one-way MOT to WHURLE conversion is therefore limited. In our future work we will be examining the ease of extending the web service by implementing a WHURLE to MOT conversion, as well as a SCORM to WHURLE, and to MOT conversion.

By moving towards a web service we aim to lay the groundwork for a middleware service that will offer an open API which AEH system designers may create a conversion extension for, which will then give them access to many other AEH systems. This work, along with additional research on the usability of authoring systems, will aim to simplify creation and re-use of AEH learning materials, thereby encouraging their use in Education and bringing the benefits of a personal educational experience to more learners

6 Acknowledgements

The research leading to this paper was supported by the European Commission under contract FP6-027026, *Knowledge Space of semantic inference for automatic annotation and retrieval of multimedia content - K-Space*, and contract 507310, the *ProLearn* Network of Excellence. Many thanks to Ilknur Celik for many useful discussions.

References

1. Alonso, G., Casati, F., Kuno, H. and Machiraju, V. (2004). *Web Services: Concepts, Architecture and Application*. Berlin: Heidelberg, Springer-Verlag.
2. Beal, V. A. (2005). *Understanding Web Services*. http://www.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp, last accessed 30th March 2006.
3. Brusilovsky, P (2001) Adaptive Educational Hypermedia. Proc. of 10th Tenth Intl PEG Conference. Tampere, Finland. pp 8-12.
4. Carro, R.M., Pulido, E., & Rodríguez, P. (2001). *TANGOW: A model for Internet based learning*. International Journal of Continuing Engineering Education and Life-Long Learning, IJCEELL, 11(1-2). Retrieved from <http://www.inderscience.com/ejournal/c/ijceell/ijceell2001/ijceell2001v11n12.html>
5. Chatterjee, S., and Webber, J. (2004). *Developing Enterprise Web Services An Architect's Guide*. New Jersey, Prentice Hall
6. Cristea, A. and De Mooij, A. (2003) *LAOS: Layered WWW AHS Authoring Model and its corresponding Algebraic Operators*. In Proceedings of the 14th Intl. World Wide Web Conference, Budapest, ACM, Hungary 20-24 May.
7. Cristea, A.I. & De Mooij, A. (2003) Adaptive Course Authoring: My Online Teacher. Proceedings of ICT'03, Papeete, French Polynesia.
8. De Bra, P. & Calvi, L. (1998). *AHA! An open adaptive hypermedia architecture*. The New Review of Hypermedia and Multimedia, 4, 115-139. London: Taylor Graham Publishers.
9. MapForce (2006) *Altova MapForce*. http://www.altova.com/download_mapforce.html, last accessed 30th March 2006.
10. Moore, A., Brailsford, T.J. and Stewart, C.D. (2001) *Personally tailored teaching in WHURLE using conditional transclusion*. Proceedings of the twelfth ACM conference on Hypertext and Hypermedia, Denmark.
11. Potts, S. and Kopack, M. (2003). *SAMS Teach Yourself Web Services*. Indiana: SAMS
12. Stewart, C., Cristea, A., Brailsford, T. & Ashman, H. (2005). 'Authoring once, Delivering many': Creating reusable Adaptive Courseware, 3rd IASTED International Conference on Web-Based Education - WBE'05, Grindelwald, Switzerland, February, 2005.
13. Stewart, C., Cristea, A., Moore, A., Brailsford, T. & Ashman, H. (2004). Authoring and Delivering Adaptive Courseware, International Workshop on Authoring of Adaptive and Adaptable Educational Hypermedia, AH2004, Workshop proceedings (part II), Eindhoven, Netherlands, August 2004.
14. WSDL, (2001). *Web Services Description Language (WSDL) 1.1*. W3C, <http://www.w3.org/TR/wsdl>, last accessed 30th March 2006
15. WSG, (2004). *Web Services Glossary*. W3C, <http://www.w3.org/TR/ws-gloss/>, last accessed 30th March 2006