Integrating the Educational Enterprise

Tracy Baving
tbaving@cs.uct.ac.za

Donald Cook
dc@cs.uct.ac.za

Trevor Green
tgreen@cs.uct.ac.za

Technical Report CS03-11-00
Department of Computer Science
University of Cape Town

Abstract

Learning management systems are being used in today's educational climate without much thought to current pedagogies and interoperability. The Open Knowledge Initiative (OKI) provides a set of Open Service Interface Definitions (OSID's) that address interoperability in the educational enterprise while remaining pedagogically flexible. Web Services are suggested as a suitable middleware to use when integrating the enterprise services using OKI. A possible integration strategy is suggested for the learning management system WebStation^3 to demonstrate OKI.

1 Introduction

The Department of Computer Science at the University of Cape Town uses the web to supplement its teaching. For every course offered by the department, a web site is put up and managed, and items such as the PowerPoint slides of the lectures, the description of the assignments, the hand in page, etc. need to be put onto each site. A few teaching assistants are hired, and they maintain these websites as part of their work for the department.

Over time, several different looking websites have been developed. This is because no instance of a course website catered fully for every course. It was more convenient to develop a new website or make use of existing third-party tools to provide the needed functionality than to take an existing instance and further develop it. This approach certainly worked in the short term, but it has resulted in a web presence that has a sloppy appearance. Furthermore, the problem of poor maintainability is now worse, and some significant bugs still exist.

The department has recently decided to obtain accreditation for its degrees from the British Computer Society (BCS). As part of the agreement, it has committed to mark assignments and return the results to the students within two weeks of assignment's deadline. The existing websites need to incorporate functionality that will assist the process of marking and feedback, and the existing practice of duplicate data entry needs to be eliminated. Instead of incorporating this functionality into the existing systems, a new and integrated system should be created. We decided to call this new system WebStation^3.

1.1 Objectives

The aims of WebStation^3 include the following:

- Integrate and improve on current software and processes in existence in the Computer Science Department at UCT.
- Automate and increase automation in processes in operation in the Computer Science Department at UCT.
- Develop a self-contained Learning Management System (LMS) that builds on the strengths and improved design limitations of other LMS's available.
- Ensure that the implementation is modular and extensible
- Focus on user requirements

The project was divided into 2 parts. One part focused on the creation and configuration of content (WebStation^3). The other part focused on providing administrative services for the entities used in WebStation^3 , as well as capture and integrate services and processes already in operation in the CS Department (AdminStation). The whole system served to function as a stand-alone, self-contained Learning Management System that met the requirements of the Computer Science Department at UCT.

Before implementing our system, research into the field of online education systems was conducted. User interviews were also conducted to obtain information about the services and systems that were currently available in the department, as well as to elicit requirements.

We implemented the system using the Microsoft.Net Framework, making use of useful features provided by ASP.Net and ADO.Net architectures for creating server-based web-applications.

1.2 Findings

Near the end of the implementation phase, two conclusions were reached. Firstly, many LMS's have been developed without an educational goal in mind. The current educational climate at the moment supports a social constructivist model for providing education. This model defines learning as a reflective process that can be enhanced through actively applying knowledge and the social collaboration of ideas. Of the LMS's available that do have some educational structure in mind, very few support this modern view on the educative process in providing an infrastructure that supports this type of learning. Pedagogy is an important aspect to consider when eliciting requirements for a LMS.

Secondly, we realised that the application programming interfaces being developed by the Open Knowledge Initiative (O.K.I.) would fully integrate the University's information technology solution. Learning management systems have evolved far, but they have each evolved independently and despite the similarities in their feature sets, no interoperability exists between them. Furthermore, they do not integrate with other third-party systems. O.K.I. is being developed to address these issues and has the potential to integrate an entire campus's IT infrastructure and even integrate the IT systems of several campuses. This paper will look briefly at what the O.K.I. is. Implementing the O.K.I. will also be discussed, with particular reference to WebStation[3].

2 What is the O.K.I.?

The Open Knowledge Initiative (O.K.I.) is a project conducted by the Massachusetts Institute of Technology. The OKI aims to "define a set of fundamental services that components of the educational environment use to work together, as well as with other enterprise applications" [Open Knowledge Initiative 2002a]. The project regards the learning management service as an enterprise service, and therefore needs to interoperate amongst other enterprise services within the organisation.

A distinctive property of the OKI is that it is architectural in nature [Open Knowledge Initiative 2002a]. The interface methods provided by the OKI integrate three general categories of software, namely the various learning applications (such as quizzing, discussion forums, etc.), the central administrative system and the academic systems (such as the library information system and digital repositories). Once this architecture is fully adopted by the education market, new components can be plugged into the educational infrastructure using OKI's API's. While the initial cost of setting up a learning technology system is not necessarily reduced, the intention is to reduce the ongoing cost of maintaining and updating the system.

The OKI defines twelve API's, which they call OKI Service Interface Definitions (OSID's), namely authorization, authentication, DBC, filing, dictionary, logging, shared, workflow, scheduling, SQL and user messaging [Thorne et al. 2002]. The API's are described abstractly and written in Java. The O.K.I. "keeps open the future possibility of expressing these services [or APIs] as interfaces in other object orientated languages such as C++ and C-Sharp. It also makes it possible to support service-based implementations using tools such as J2EE and .NET."

The APIs are divided into two groups: the Common Service APIs and the Educational Service APIs. The Common Service APIs define APIs that are not specifically educational, such as authorization, authentication, etc. The Educational Service APIs are specifically educational. Examples of Educational Service APIs are class administration, assessment, communication services, etc. A summary of all the APIs are presented in [Thorne et al. 2002].

O.K.I. is specifically designed for the needs of higher education and makes use of existing standards (e.g. SCORM, IMS, etc.) wherever appropriate.

Kumar *et al.* [Vijay Kumar et al. 2001] argues that educational architectures need to fulfill three requirements. Firstly, "The architecture of learning management systems must support the development of diverse, customised tools in the support of disciplined or pedagogically specific needs." In other words, the architecture does not limit the institution to one centralised, monolithic system, but allows smaller entities within the institution to create their own customised solution to suit their pedagogical needs. For example, if a institution prefers WebCT as their learning management system, but it is not pedagogically sufficient for a particular entity within that institution, then that entity should be able to develop or choose a system that satisfies their pedagogical requirements while integrating with the institution's IT infrastructure.In U.C.T's context, this means that WebCT may be used at the university while the Computer Science Department uses WebStation[3].

Secondly, the architecture must be an "architecture that endures". There is no way of knowing what kinds of devices, operating systems, protocols, etc. will be developed over the next few years. The architecture should survive technological advancements such as these, therefore it is necessary to keep the design of the learning system as technologically neutral as possible. "The enterprise approach is one of sustainability and scalability. It calls for the development of an infrastructure that encourages a wide variety of academic applications sharing data and services via open communication protocols and open programmer application interfaces." [Vijay Kumar et al. 2001]

Thirdly, the architecture must be widely adopted. The OKI is a collaborative project led by the M.I.T. and supported by Stanford University, Dartmouth College, Harvard University, North Carolina State University, the University of Michigan, the University of Pennsylvania, the University of Wisconsin-Madison, the University of Washington and the University of Cambridge.

Kumar further argues [Vijay Kumar et al. 2001] that the OKI's strength lies in its modularity. OKI's Architectural Overview [Thorne et al. 2002] states their modularity goals as follows:

"The OKI architecture must accommodate diverse environments and growth in both technology and pedagogy. It must isolate the courseware tools from infrastructure services. It must also isolate one courseware component from changes in another courseware component. It must scale to handle the demands of large institutions with many students, faculty, and courses on multiple campuses, or nonstandard courses and participants."

3 Web Services - The Middleware

Having interface definitions such as those proposed by the OKI is not sufficient to integrate the various services. The various service definitions need to communicate with one another in spite of the language, operating system or architecture that each of them are implemented on. Another consideration is the current network architecture of the institution. The institution should not have to fiddle too much with the firewall configurations or network architecture.

Many middleware solutions exist to provide a distributed solution (such as CORBA, COM+ or RMI), but Web Services offers the simplest solution [Stal 2002]. Two advantages of Web Services over other middleware solutions in terms of interoperability are that they are language agnostic and based on web technologies.

Web Services add a new level of interoperability through a SOAP layer, which separates the transport protocol from the Web Service payload. This allows Web Services to be requested over various transport protocols, including HTTP, RMI/IIOP, Instant Messaging and IBM WebSphere MQSeries to name a few [Fremantle et al. 2002].

Typically Web Services are implemented over HTTP, which allows services to be requested over the Internet, and not only within a intranet. This enables a simple distributed solution to institutions that have many campuses. Furthermore, if each campus has its own firewall and allows Internet access, then no added configuration is necessary to allow Web Services.

In addition to its simplicity, Web Services have a myriad of protocols that complement it. The protocols that concern interoperability particularly are the Web Service Description Language (WSDL); the Universal Description, Discovery and Integration of Web Services (UDDI); and the Web Service Inspection Language (WSIL).

3.1 WSDL

The Web Services Description Language allows for the description of the inputs and outputs of a Web Service. It allows the server to publish the interface of a Web Service so that if a client sends a SOAP request in the correct format as described by the WSDL instance, an appropriate SOAP response will be returned in the correct format as described by the same WSDL instance.

According to [Fremantle et al. 2002], WSDL has two strengths:

1. It enforces separation between interface and implementation.

   Fremantle et al. writes: "The interface must be described as an abstract PortType, which is defined in terms of the input and/or output messages that it supports for each operation. This abstract service is then bound to a particular implementation at a particular location using a Port (location) and Binding (implementation style)." This abstract description of interfaces permits Web Services to have the degree of language agnosticism that it has.

2. It is inherently extensible.

   Fremantle continues: "The core WSDL specification only describes the abstract interface and the structure of ports and bindings. Actual implementation types, such as SOAP, are described using extensions. This extensibility means WSDL can be used to describe almost any service-orientated interaction."

Various tools exist which make use of WSDL to generate client proxy stubs. These stubs do not necessarily have to be in the same implementation language as Web Service's implementation language.

3.2 UDDI

UDDI is a standard maintained by Oasis (http://www.oasis-open.org/). Companies such as Microsoft and IBM host UDDI implementations where businesses may publish their services. UDDI may also be implemented privately.

Educational institutions may make use of UDDI to publish services to other institutions. An example of such a service would be a digital library of publications. Congia, et al [Congia et al. 2003] have implemented a SOAP-based protocol for OAI-PMH, which is a first step toward having Web Services implemented that can access open archives. UDDI can publish the availability of these services to a central location so that they may be discovered and utilised.

3.3 WSIL

WSIL is a complement to UDDI. UDDI offers discovery of services on a global or wide-scale. WSIL concentrates on a particular site and discovers the services available locally. The list of services returned by a WSIL query includes either a link to each service's WSDL or a link to a UDDI service entry. WSIL provides institutions with a lightweight alternative to UDDI when the locality of the services of known.

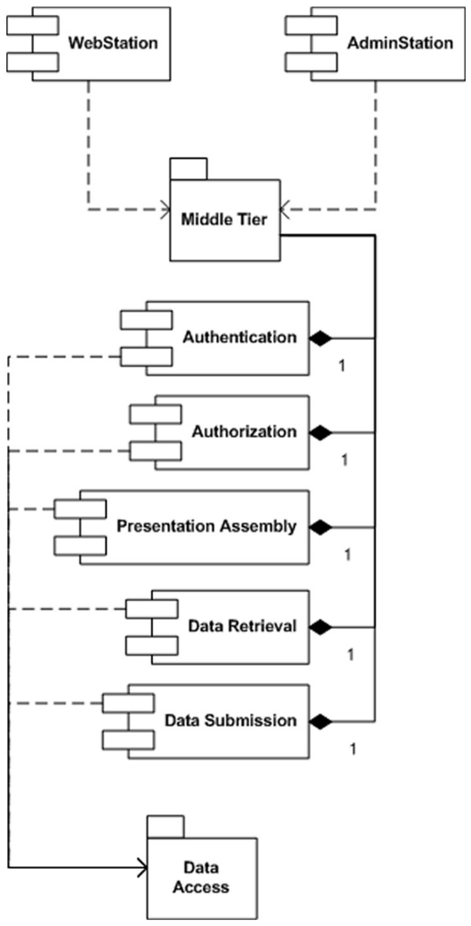

Figure 1: Current WebStation^3 architecture

4 Implementing OKI in WebStation^3

Both WebStation and AdminStation are built using a 3-tiered architecture. The services provided by the middle tier are authentication, authorization, presentation assembly, data submission, data retrieval, calendar, tutorial management, logging and Document Storage. The arrangement of these services are shown in figure 1 and the implementation after integration using OKI is shown in 2.

Authentication and authorization have their obvious place in the new OKI based architecture.

The course calendar component (not shown) will use two of the OSID's, namely Scheduling and Hierarchy. Each course has its own calendar in the old architecture. However, there should be the ability to have events that are common across several courses. This is where the Hierarchy OSID comes in, because it allows a course to "inherit" a calendar from another course, or even inherit calendars that do not belong to a course such as a university's academic calendar. Desktop organisers such as Microsoft Outlook can also be configured to use the OKI OSID's (and this configuration needs to only be written once for each software package) and then these desktop organisers may be used to manipulate course calendars.

The Discussion Board and Announcements components will integrate with the User Messaging OSID. Messages can

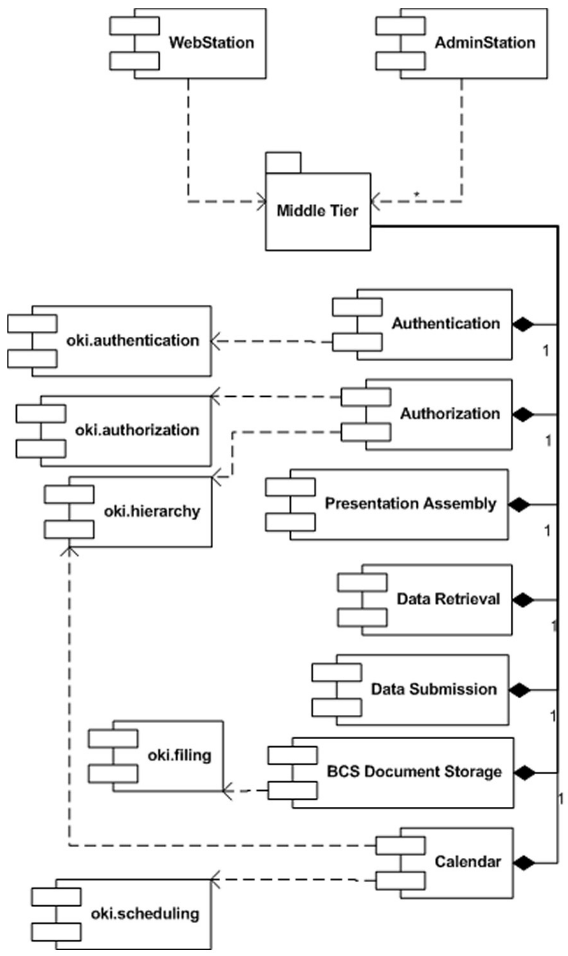

Figure 2: WebStation^3 architecture using OKI

be organised into groups and threads by implementing them as nodes in a hierarchy as provided by the Hierarchy Service Interface Definition. Discussion forums may even be linked to an email address using the OKI OSID's and sending a message to this email address is equivalent to posting to the discussion forum. The back-end of the scheduling OSID can even link with a SMTP server and email the students in the course whenever a message is posted to that group. The possibility even exists of sharing discussion groups across learning management systems.

The BCS Document Storage service supplied by the WebStation^3 middle tier is an ideal candidate to wrap the OKI Filing OSID because the two services provide exactly the same functionality.

Finally, the logging service provided by the WebStation^3 middle tier (not shown) can wrap the OKI Logging OSID. The back-end of the Logging OSID can perform on-the-fly processing of logging information it receives and present an analysed view of the logs through a second implementation of the Logging OSID since The Logging OSID makes provision for read-only logs and write-only logs.

5 Conclusion

Learning management systems are predominantly pedagogy specific and therefore restrict their lifetime. WebStation^3 attempts to be pedagogically neutral, but this may not necessarily be the case. The Open Knowledge Initiative provides an architecture that allows the learning management system to be easily disposable without affect the majority of a institutions information technology infrastructure. Institutions will therefore no longer have to be pedagogically restricted by their information technology implementations.

The Open Knowledge Initiative, coupled with Web Services, provides a very flexible distributed model. Built on web standards and SOAP, Web Services offer a language agnostic, platform agnostic remote procedure call mechanism that can integrate the institution's enterprise services and integrate services between institutions.

References

Congia, S., Gaylord, M., and Merchant, B. 2003. Soapifying the open archives. Available from http://pubs.cs.uct.ac.za/.

Fremantle, P., Weerawarana, S., and Khalaf, R. 2002. Enterprise services. *Communications of the ACM 45*, 10 (October), 77–82.

Hou, J., and Zhang, Y. 2002. Constructing good quality web page communities. In *Proceedings of the thirteenth Australasian conference on Database technologies*, Australian Computer Society, Inc., 65–74.

Open Knowledge Initiative, 2002. Oki product description introduction. Available from http://web.mit.edu/oki.

Open Knowledge Initiative, 2002. What is the open knowledge initiative? Available from http://web.mit.edu/oki.

Stal, M. 2002. Web services: Beyond component-based computing. *Communications of the ACM 45*, 10 (October), 71–76.

Talbott, D., Gibson, M., and Skublics, S. 2002. A collaborative methodology for the rapid development and delivery of online courses. In *Proceedings of the 20th annual international conference on Computer documentation*, ACM Press, 216–225.

The ADL Initiative. The sharable content object reference model (scorm) specification 1.2. Available from http://www.adlnet.org/.

The IMS Global Consortuim. The ims global consortium standards. Available from http://ww.imsglobal.org/.

Thorne, S., Shubert, C., and Merriman, J., 2002. Oki architecture overview. Available from http://web.mit.edu/oki/.

Vijay Kumar, M. S., Marriman, J., and Long, P. D. 2001. Building "open" frameworks for education. *EDUCAUSE* (November).

World Wide Web Consortium, 2003. Web services architecture (working draft). Available from http://www.w3.org/TR/2003/WD-ws-arch-20030808/.