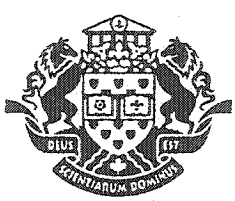



Université d'Ottawa · University of Ottawa

# Université d'Ottawa · University of Ottawa

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

Zhengfang XU

AUTEUR DE LA THÈSE - AUTHOR OF THESIS

M.Sc. (System Science)

GRADE - DEGREE

Systems Science

FACULTÉ, ÉCOLE, DÉPARTEMENT - FACULTY, SCHOOL, DEPARTMENT

TITRE DE LA THÈSE - TITLE OF THE THESIS

A Web Oriented Framework for Distributed E-Learning

A. El Saddik

DIRECTEUR DE LA THÈSE - THESIS SUPERVISOR

CO-DIRECTEUR DE LA THÈSE - THESIS CO-SUPERVISOR

EXAMINATEURS DE LA THÈSE - THESIS EXAMINERS

L. Peyton

J. Zhao

J.-M. De Koninck, Ph.D.

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES

SIGNATURE

DEAN OF THE FACULTY OF GRADUATE AND POSTDOCTORAL STUDIES

# A Web Oriented Framework for Distributed E-Learning

ZHENGFANG XU

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of the requirements
for the degree of Master of System Science

School of Management
School of Information Technology and Engineering
University of Ottawa

*Your file Votre référence*
*ISBN: 0-612-90359-1*
*Our file Notre référence*
*ISBN: 0-612-90359-1*

# Acknowledgements

I would like to express my sincere thanks to my supervisor, Professor Abdulmotaleb El Saddik, for his valuable advice, encouragement, patience and effort on my thesis. I feel very fortunate to have had his supervision and extremely grateful for the learning experience he has given me.

I would also like to extend my sincere thanks to the faculty members of the Multimedia Communication Research Laboratory (MCRLab) group for their help and support.

Last, but not least, I am grateful to my family for their encouragement and support through the years.

# Abstract

The objective of this thesis is to propose a Web services oriented framework for distributed e-learning systems aimed at providing a flexible integration model in which all the learning components and applications are well defined, effectively discovered and loosely connected. Web services provide an essential deploy environment to realize dynamic e-learning/e-business systems by facilitating application-to-application interaction. Using the proposed framework, learning service providers will be able to publish their learning objects or services universally and learning service requesters can retrieve those services anywhere, any time with any device (wired or wireless) through common communication protocols. The key values of interoperability and accessibility in the proposed architecture enhance the future distributed e-learning systems to communicate more efficiently and share data more easily. A proof of concept of the proposed system is designed and implemented in a J2EE (Java 2 Enterprise Edition) combined with J2ME (Java 2 Micro Edition) environment, using JAX Pack (Java for XML Pack) for building essential Web services and kSOAP package for parsing SOAP (Simple Object Access Protocol) messages on lightweight platforms. The implementation is a successful demonstration that learning services can be easily accessed through standard Web services interface. A cross-platform service invocation (C# to Java, Windows to Linux) is successfully accomplished in this implementation.

# Table of Contents

# List of Figures

# List of Tables

# List of Listings

# Glossary of Terms and Symbols

| | |
|---|---|
| ADL | Advanced Distributed Learning |
| AICC | Aviation Industry CBT Committee |
| API | Application Programming Interface |
| ARIADNE | Alliance of Remote Instructional Authoring & Distribution Networks for Europe |
| ASF | Apache Software Foundation |
| ASP.NET | Active Server Page .NET |
| AXIS | Apache extensible Interaction System |
| CBI | Computer-Based Instruction |
| CDC | Connected Device Configuration |
| CLDC | Connected Limited Device Configuration |
| CORBA | Common Object Request Broker Architecture |
| DCOM | Distributed Component Object Model |
| DOM | Document Object Model |
| DSL | Digital Subscriber Line |
| EJB | Enterprise Java Beans |
| ERP | Enterprise Resource Planning |
| FP | Foundation Profile |
| GUI | Graphical User Interface |
| HTTP | Hypertext Transfer Protocol |
| IDL | Interface Definition Language |
| IEEE | Institute of Electrical and Electronics Engineers |
| IIOP | Internet Inter-ORB Protocol |
| IIS | Internet Information Server |
| ITS | Intelligent Tutoring Systems |
| J2EE | Java 2 Platform, Enterprise Edition |
| J2ME | Java 2 Platform, Micro Edition |
| J2SE | Java 2 Platform, Standard Edition |
| JAX | Java APIs for XML |
| JAXM | Java API for XML Messaging |
| JAXP | Java API for XML Processing |
| JAXR | Java API for XML Registries |
| JAX-RPC | Java API for XML-based RPC |
| JDBC | Java Data Base Connection |
| JMS | Java Message Service |
| JNDI | Java Naming and Directory Interface |
| JRMP | Java Remote Method Protocol |
| JVM | Java Virtual Machine |
| JWSDP | Java Web Services Development Package |
| KPS | Knowledge Pool System |
| LAN | Local Area Network |
| LMS | Learning Management System |
| LOM | Learning Object Metadata |
| LTSC | Learning Technology Standards Committee |

| | |
|---|---|
| MIDP | Mobile Information Device Profile |
| MOM | Message Oriented Middleware |
| MVC | Model-View-Control |
| NAICS | North America Industry Classification System |
| OASIS | Organization for the Advancement of Structured Information Standards |
| OMG | Object Management Group |
| ORB | Object Request Broker |
| QoS | Quality of Service |
| RMI | Remote Method Invocation |
| RPC | Remote Procedure Call |
| SAX | Simple API for XML Parsing |
| SCO | Sharable Content Objects |
| SCORM | Sharable Content Object Reference Model |
| SMTP | Simple Mail Transport Protocol |
| SOAP | Simple Object Access Protocol |
| UDDI | Universal Description Discovery and Integration |
| UML | Unified Modeling Language |
| URL | Uniform Resource Locater |
| URI | Uniform Resource Identifier |
| UUID | Universally Unique ID |
| W3C | World Wide Web Consortium |
| WSDL | Web Services Description Language |
| XML | eXtended Markup Language |
| XSD | XML Schema Datatypes |

# Chapter 1 Introduction

## 1.1 Motivations

The Internet has had a revolutionary effect on how organizations conduct their business, opening up new possibilities for communicating with employees, customers and partners. e-business has raised the stakes for organizations, greatly increasing the working process speed at which they are expected to compete. E-commerce brings efficiencies to business-to-business interaction by means of aggregation of services and its potential to reach large numbers of potential business partners. One of the important aspects to bringing efficiencies to business-to-business interaction is the need for companies to integrate their applications with those of their partners in the marketplace. However, without industry standards for integration it is very difficult to achieve efficiencies and flexibilities in the processes between businesses [1]. The traditional, tightly coupled distributed communication model between an enterprise and its partners can no longer meet the future requirements of e-commerce or e-business. It is not flexible and interoperable. Recent emergence of Web services model is an attempt to provide an answer to these problems. Web services are a set of open standards that facilitate application-to-application interaction in a systematic way. Web services model provides a state of the art solution to deal with the heterogeneity of Web-based distributed system.

Just as the Internet has allowed companies to capture the benefits of e-business or e-commerce, it is now rapidly transforming the way we approach distributed e-learning systems. From CBI (Computer-Based Instruction) to ITS (Intelligent Tutoring Systems), e-learning system has come a long way. The growth of Internet has provided unanticipated and unexpected ways for teaching and learning at distances. Many e-learning providers have designed some sophisticated Web based solutions for distributed learning, which contain learning content management and delivery, course management, user management, collaboration and communication, etc. However, most of these accomplishments are based on the Web's fundamental nature as a medium to enable human-to-application interactions. They also have the similar problems as most businesses do. A question arise: What kind of solution could raise e-learning systems into

a higher stage supporting application-to-application (learning system-to-learning system) communication dynamically regardless of which application platforms are being relied on and what programming language are used to develop? There is a need for a standard mechanism for supporting complete automation through all aspects of end-to-end learning process that includes finding suitable learning contents, learning objects or learning management services, getting information about their services and invoking their services. It is said that Web services give a kind of technology that if properly used the answer of the above stated problem can be found. This thesis is trying to research this possibility.

*"A Web service is a software system identified by a Uniform Resource Identifiers (URI), whose public interfaces and bindings are defined and described using XML. Its definition can be discovered by other software systems. These systems may then interact with the Web service in a manner prescribed by its definition, using XML based messages conveyed by Internet protocols."* – World Wide Web Consortium (W3C) [2]

Web services are standards for describing, publishing, discovering and binding application interfaces. The set of standards include a standard specification for public registries known as Universal Description Discovery and Integration (UDDI), a description language namely Web Services Description Language (WSDL), a distributed object communication protocol called Simple Object Access Protocol (SOAP) and a dynamic, self-defining information specification Language with semantic support known as eXtended Markup Language (XML).

## 1.2 Contributions

In this thesis project, a new framework is proposed, which is not only Web based but also Web services oriented. It provides a flexible integration model to achieve the emerging needs of distributed e-learning systems. The framework facilitates distributed e-learning systems by providing a comprehensive platform in which all the shareable learning components are published, described, located and invoked in a standardized way. Also the framework is explained with two typical scenarios: single sign on process and learning content aggregation for being further comprehended. The major contributions related to the proposed framework are briefly summarized in the following:

- The thesis describes the details of how a distributed e-learning system based on this framework is analyzed and designed using Unified Modeling Language (UML).
- Based on the analysis and design, a Web services oriented distributed e-learning system prototype called UbiLearn is developed at the Multimedia Communication Research Laboratory at the University of Ottawa, Canada.
- The essential implementation technologies for building Web services like Java APIs for XML (JAX) are depicted.
- Proving the platform independency of Web services: a service request client written in C# running on a Windows XP server successfully invokes a Web service developed by JAX that resides on a Linux server.
- The essential technologies for wireless devices to access Web services like kSOAP are also described.
- The performance and scalability characteristics of the proposed architecture are evaluated by sending several SOAP requests concurrently.
- The needed software, the steps of setting up a Java-enabled Web services environment and the installation guide for UbiLearn are summarized.
- Publications:
    - Zhengfang Xu, Zheng Yin and Abdulmotaleb El Saddik. "A Web Services Oriented Framework for Wired and Wireless E-Learning Systems". In Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE 2003) Montreal, Quebec, Canada, May 4-7, 2003.
    - Zheng Yin, Zhengfang Xu and Abdulmotaleb El Saddik. "Study of Metadata for Advanced Multimedia Learning Objects". In Proceedings of the Canadian Conference on Electrical and Computer Engineering (CCECE 2003) Montreal, Quebec, Canada, May 4-7, 2003.

## 1.3 Organization

Chapter 2 presents the traditional middleware used in distributed computing systems, such as, Remote Procedure Call (RPC), Message Oriented Middleware (MOM) and Common Object Request Broker Architecture (CORBA). Then the general architecture

and standard-based protocols of the new emergence of Web services are described in detail.

Chapter 3 describes the infrastructure of distributed e-learning. First a virtual environment called Share Place in which users can use content and tools for collaborative learning is described. Then two related works of distributed e-learning: ARIADNE system and SCORM are introduced.

Chapter 4 presents the details of the Web services oriented framework for distributed e-learning. The system architecture, the analysis and design models for the Web services-based e-learning system are discussed.

Chapter 5 describes the implementation details of the e-learning system enhanced by Web services: UbiLearn. The cross-platform service invocation is given. And the essential technologies for wireless devices to access Web services are presented. This chapter is ended with some useful tools for monitoring and generating Web services entities.

Chapter 6 presents the methodology for scalability and performance testing and analyzes the test results of a specific Web service to show how well the Web service serves clients under different loading conditions.

Chapter 7 gives the conclusions of the thesis and suggests directions for future research.

# Chapter 2 Web Services Overview

Recent emergence of Web services provides a new distributed computing platform to show how best it can support Web based application-to-application communications in a systematic way. Before we discuss the details of Web services, we will first briefly describe some traditional distributed computing methodologies to show why Web services are a natural consequence of the evolution of the Web.

## 2.1 Traditional Distributed Computing

Distributed computing system is a kind of computer system in which several interconnected and relatively independent sub systems share the computing tasks and resources through some common communication middleware over a network. [49] The common communication middleware used in distributed environments are categorized in three types: i) Remote Procedure Call (RPC) (e.g. Java's RMI – Remote Method Invocation, Microsoft's DCOM – Distributed Component Object Model), ii) Message Oriented Middleware (MOM) (e.g. Java's JMS – Java Message Service, IBM's MQ Series, Microsoft's MSMQ – Microsoft Message Queuing), iii) Middleware framework (e.g. CORBA – Common Object Request Broker Architecture). We review these middleware in this section.

### 2.1.1 Remote Procedure Call (RPC)

RPC is appropriate for client/server[1] applications in which the client can send a request and wait for the server's response before continuing its own processing. The most common RPC implementations are Java's RMI [3] and Microsoft's DCOM [4].

#### 2.1.1.1 Remote Method Invocation (RMI)

RMI allows object-to-object communication between different Java Virtual Machines (JVM). The RMI architecture of three-abstraction layer over TCP/IP makes remote objects in separate JVMs look and act like local objects. The three layers are:

---

[1] The term "client" does not refer to a physical client machine but actually refers to the application invoking a remote procedure on a remote object. Similarly, the term "server" does not refer to a physical server machine but a single remote object with methods that can be remotely invoked.

- The Stubs/Skeletons Layer

  The stub is a client-side object that represents the remote object. The stub plays a role of a proxy for the remote service implementation. The skeleton layer help the object communicate with the stub across the RMI link. The skeleton reads the parameters from the link, makes the call to the remote object, accepts the return value, and then passes the return value back to stub.

- The Remote Reference Layer

  This layer handles the details relating to interpreting and managing references made by clients to the remote objects. This layer provides a JRMP (Java Remote Method Protocol), which is a specific object that represents a handle to the remote object.

- The Transport Layer

  This layer is based on TCP/IP connections between machines in a network. It is responsible for setting and managing those connections between the JVMs.

An RMI distributed application uses an RMI registry, which is one of the naming services provided by the Java Naming and Directory Interface (JNDI) to obtain a reference to a remote object by looking up the name that objects are associated with. The lookup returns a remote reference, a stub, to the object.

#### 2.1.1.2 Distributed Component Object Model (DCOM)

DCOM is based on the original Distributed Computing Environment (DCE) [5] standard Remote Procedure Call (RPC) infrastructure and was created as an extension of Component Object Model (COM) to allow the creation of server objects on remote machines. COM defines a binary structure for the interface between the client and the object using Microsoft Interface Definition Language (IDL). Just like Java's RMI, there are also three layers in DCOM architecture: the Proxy/Stub layer, the COM Library layer and the RPC Runtime layer. The client calls the interfaces of the remote object through the "proxy", which marshals the parameters and passes them into the remote server "stub". The "stub" unmarshals the parameters and makes the actual call inside the server object through the COM Library channel. When the call completes, the "stub" marshals return values and passes them to the "proxy", which in turn returns to them to the client.

The low-level network communication between client and server is accomplished through the RPC Runtime layer.

All COM objects are registered with a component database server called Microsoft Transaction Server (MTS), where a client can invoke a COM API to instantiate a new COM object and get an interface pointer at the object.

From the above introduction, RPC reduced the complexity involved in the development of distributed processing by allowing the remote component to be accessed without knowledge of the low-level network information, but there are two major limitations for the RPC mechanism.

- RPC implementations are usually incompatible with other RPC implementations. It is almost impossible to directly invoke a remote COM object with a Java RMI client.
- RPC uses a synchronous request-reply mechanism that requires that the client and server are always available and functioning.

### 2.1.2 Message Oriented Middleware (MOM)

MOM is a kind of middleware that resides in both client and server portions of a distributed architecture and typically supports asynchronous calls between the client and server applications by exchanging messages. [7] A message may be a request, a report, or an event sent from one part of an application to another. The Java Message Service (JMS) [6] is a Java API that allows application to create, send, receive and read messages.

#### 2.1.2.1 Java Message Service (JMS)

A JMS application is usually composed of these parts: a JMS provider, a JMS client, messages and a service registry like JNDI. The JMS offers the two most common models for messaging:

- Publish/Subscribe

  Publish/Subscribe is a one-to-many publishing model where client applications publish message to topics, which are in turn subscribed to by other clients that are interested in those topics.
- Point-to-Point

Point-to-Point provides a traditional queuing mechanism where a client application sends messages, through a queue, to one receiving client that obtains the messages sequentially.

Since a message can contain formatted data, method invocation requests, or routing information, MOM abstracts application interface into data descriptions so that it enables applications to exchange messages with other programs without having to know what platform or processor the other application resides on. Unlike RPC, MOM provides an asynchronous mechanism (most MOM implementations like JMS support both asynchronous and synchronous way) that enables the client application to send action requests without being aware of the remote object's current status. It is the messaging middleware's responsibility to notify a server that some actions need to be taken. The asynchronous mechanism makes MOM more flexible than RPC, but MOM is also implemented as a proprietary product, which means there is no consistent protocol for data exchanging among different products and the weakness of vender dependence still negatively impacts on a system's flexibility, portability and interoperability.

### 2.1.3 Middleware Framework - CORBA

CORBA – Common Object Request Broker Architecture, developed by the Object Management Group (OMG) [8], was expected to be an open, vendor-independent architecture and infrastructure that applications can use to communicate each other in a decentralized environment. A CORBA based program from any vendor, on almost any computer, operating system, programming language, and network, can interoperate with a CORBA based program from the same or another vendor, on almost any other computer, operating system, programming language, and network. [9] CORBA follows basically the same approach as RPC does but with few modifications. The CORBA framework consists of three major parts:

- The Interface Definition Language (IDL)
- The Object Request Broker (ORB)
- The Internet Inter-ORB Protocol (IIOP)

The IDL is a language created by OMG for defining interfaces of distributed objects. IDL is programming language independent, but needs IDL-to-language *mappings*, allowing an IDL interface to be mapped into a particular language.

The ORB's primary responsibility is to facilitate the creation and transmission of request and reply messages that occur between clients and objects. Clients use the object references to direct their invocations, identifying to ORB the exact instance they want to invoke.

The IIOP is a standard protocol, defined by OMG, which is used by ORBs to send messages back and forth between clients and remote objects. Any two ORB applications or products written by different programming language can communicate with each other as long as they are using this protocol.

Interoperability is accomplished at a certain level with those standards, but it is not perfect, it is only CORBA-compliant interoperability. The limitations lie in:

- The language you choose to develop CORBA component must have a mapping defined between itself and the IDL. The standardized mappings are from IDL to: C, C++, Java, COBOL, Smalltalk, Ada, Lisp, Python, and IDLscript. [9] That means you cannot write a CORBA client or object using any other languages such as C# or Visual Basic at the present.
- Though IIOP is a vendor-independent protocol, it is only for ORB implementations, which means IIOP is not a common network protocol for widely using.
- While CORBA defines a standard, there is great latitude in many of the implementation details. Users have to keep learning the different ORB features and capabilities developed by different vendors. [10]

## 2.2 Web Services – The Next Generation of Distributed Computing

As we discussed above, some technologies were developed to help realize object-to-object communication in a relatively transparent way. Some approaches have successfully accomplished the mission of applications executing within different machines to communicate each other without knowing where they are. However, most of the attempts are highly vendor dependent and their objects are tightly coupled. So, the

key components of this work are focusing on interoperability, flexibility and efficient application integration within heterogeneous distributed systems.

Web services are different from traditional distributed computing models. Web service is a new emerging solution for providing application-to-application communication over the Internet. It provides a flexible way for applications interaction within different platforms through a well-defined interface, which is described using an XML-based standard functional description language. Web services are self-contained, self-describing modular applications that can be published, located and invoked across the web.

### 2.2.1 Web Services Basic Architecture

Before we talk about the generic standard protocols of Web services, let's look at the Web services model first. There are three roles in the Web services framework: *service requester*, *service broker* and *service provider*. [11] The relationships among them is shown below:

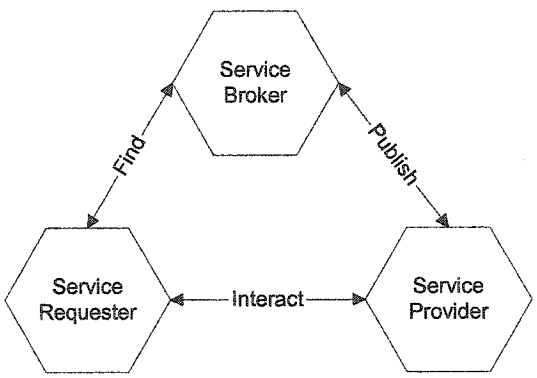



**Figure 2-1** The Web Services Basic Architecture [11]

The basic architecture includes Web services technologies capable of: [2]

- Exchanging messages
- Describing Web services
- Publishing and discovering Web services descriptions

The *service requester* is one who requests a particular service. It involves specifying important information about desired services so the service can be effectively found in a UDDI (Universal Description Discovery and Integration) registry. Then the requester can invoke the services provided by the *service provider* based on some generally available

terms such as SOAP (Simple Object Access Protocol). The *service broker* is responsible for pairing up requestors and providers. The broker provides the registry and interfaces through which requests can be made. The *service provider* is responsible for developing, implementing and publishing the services. A service description also called a WSDL (Web Services Description Language) document, which contains relevant metadata about the service for search and retrieval, should also be published in a UDDI registry by the provider.

### 2.2.2 Web Service Architecture Systematic View

The basic Web services architecture models the interactions among three roles. It gives an overall idea that how Web services are published, found and invoked. Figure 2 illustrates a systematic view for a request client that consumes application logic from a Web service.

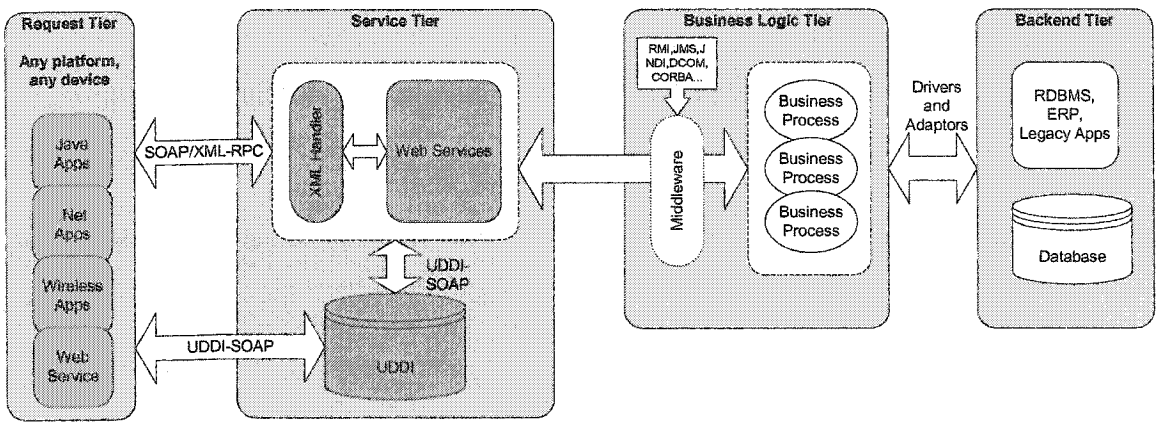



**Figure 2-2** General Four Tiers Web Services Architecture in A Systematic View

A general Web services architecture usage scenario could be divided into four tiers: *Requester tier, Service tier, Business logic tier* and *Backend tier*. As we can see from this block diagram, an independent *Service Tier* is added into a traditional client-middleware-database architecture to show how flexible a Web service can integrate an existing distributed system infrastructure.

In this Web services architecture, since SOAP is the standard data exchanging interface between the first two tiers, the *Request tie* could be any platform like wired or wireless, Unix or Windows, any kind of application like a Java Servlet or a .NET COM object or even another Web service. The request applications send SOAP messages to a

UDDI registry to retrieve the information of their desired services, then they can communicate with the Web Services directly through SOAP or XML based RPC. In the *Service tier*, a Web service application can publish its service description document called WSDL file to a UDDI registry. An XML handler is designed here to parse and compose the request and response SOAP messages. Intercepted data from request messages are routed to the Web services component, in which the desired functional objects are invoked via the underlying middleware. Then the Web services component retrieves the result from the middleware, encodes it in XML and send back the response back to the *Request tier*. The middleware that resides in the *Business logic tier* could be any kind of sophisticated technologies such as RMI, JMS, DCOM, CORBA or any other vendor specified API. The business/application process components are actually the requesters ultimately interested in accessing. The last *Backend tier,* the resources that the requester is ultimately trying to access could be a relational database system, an ERP system or a legacy mainframe system.

No matter how complicated the existing system infrastructure is, the most critical thing for your system to be Web services oriented is how to build the *Services Tier* facing to the potential business partners and seamlessly integrate with your system. The details of how to design and implement this tier are described in Chapter 4 and Chapter 5, and now we will see how the Web services standard protocols provide a level of interoperability, loosely coupled communication mechanism.

## 2.3 Web Services Standard-Based Technologies and Protocols

The Web services architecture is based on the following standard-based technologies and protocols:

### 2.3.1 XML (eXtensible Markup Language)

XML is a commonly used language for Internet-ready documents and development. It is the root technology of Web services for some key reasons: First, it provides a common syntactical way of expressing structured data in the form of human readable documents. Second, XML is an application-neutral data exchange, which means XML is same for Java platform, .NET platform and other possible platform. The XML structure is easily

navigated by third party applications. Third, XML namespaces enable applications to share semantic elements contained within a particular document. All these above are necessary to develop system implementations in a neutral manner.

As mentioned before, XML is not a binary format. It is composed of XML identified tags, which sometimes give you the semantics (meaning) of the data. With the hierarchical structure of XML document, we can access the data part we are interested in easier and faster. The following is a simple example of an XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<CourseList>
    <course>
        <courseID>CSI5111</courseID>
        <name>Web Services</name>
     </course>
< CourseList>
```

**Listing 2-1** A Simple XML Example

### 2.3.2 SOAP (Simple Object Access Protocol)

SOAP is the protocol of choice for Web service users to use to transport messages between Web service entities. SOAP is a lightweight protocol based on XML for exchange of information in a decentralized and distributed environment. It rides atop lower-level transport protocol such as HTTP (Hypertext Transfer Protocol), SMTP (Simple Mail Transport Protocol), which, in turn, rides atop a true transport protocol such as TCP/IP. HTTP, SMTP are now the most common protocols for data exchanging over the Internet.

Communication application using SOAP messaging can be implemented in a stateless manner since SOAP provides one-way messaging, which is the same asynchronous mechanism as MOM uses but it is more interoperable because SOAP is the consistent protocol among different service vendors. Another goal of SOAP is to use XML to enable RPC over HTTP, which means RPC invocations do not suffer from vendor dependence any more (RMI relies on JVM, DCOM relies on Windows platform, only CORBA compliant objects can invoke ORB over IIOP) because HTTP is the standard communication protocol over the Internet.

A SOAP message contains two SOAP-specific sub elements within the overall `Envelope` element, namely a `Header` element and a `Body` element. [12] Here is an example:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/">
    <soap-env:Header/>
    <soap-env:Body>
        <CourseList>
          <course>
              <courseID>CSI5111</courseID>
              <name>Web Services</name>
          </course>
        <CourseList>
    </soap-env:Body>
</soap-env:Envelope>
```

**Listing 2-2** A Simple SOAP Example

SOAP has been released version 1.2 by the World Wide Web Consortium (W3C). [13]

### 2.3.3 WSDL (Web Services Definition Language)

WSDL provides the template through which Web services are described and subsequently published. WSDL is a standard XML-formatted language that defines the ways that any potential client can contact and invoke a Web service. It defines an abstract interface and bindings to a particular message format. It also defines how to locate a service (e.g. Uniform Resource Locater (URL)).

A WSDL document uses the following elements in the definition of a Web service: [14]

- Types – a container for data type definitions using some type system (such as XML Schema Datatypes (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.
- Service – a collection of related endpoints.

Here is an example of a WSDL file:

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="Opposite"
targetNamespace="http://opposite.org/wsdl"
xmlns:tns="http://opposite.org/wsdl"
xmlns="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/">
<types/>
<message name="OppositeIF_checkRemoteUser">
    <part name="String_1" type="xsd:string"/>
    <part name="String_2" type="xsd:string"/>
</message>
<message name="OppositeIF_checkRemoteUserResponse">
    <part name="result" type="xsd:int"/>
</message>
<portType name="OppositeIF">
    <operation name="checkRemoteUser" parameterOrder="String_1
String_2">
      <input message="tns:OppositeIF_checkRemoteUser"/>
      <output
message="tns:OppositeIF_checkRemoteUserResponse"/>
    </operation>
</portType>
<binding name="OppositeIFBinding" type="tns:OppositeIF">
    <operation name="checkRemoteUser">
      <input>
        <soap:body
```

```
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" namespace="http://opposite.org/wsdl"/>
      </input>
      <output>
        <soap:body
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
use="encoded" namespace="http://opposite.org/wsdl"/>
       </output>
   </operation>
</binding>
<service name="Opposite">
    <port name="OppositeIFPort"
binding="tns:OppositeIFBinding">
      <soap:address
location="http://137.122.109.39:8080/jaxrpc-
service/jaxrpc/OppositeIF"/>
    </port>
 </service>
</definitions>
```

**Listing 2-3** A WSDL File Example

WSDL has been released version 1.2 by the W3C. [14]

### 2.3.4 UDDI (Universal Description Discovery and Integration)

Traditional distributed computing methodologies like RMI, DCOM and CORBA, they have their own naming and registry service for locating desired remote objects. However, those services are completely vendor dependent. A Java Naming and Directory Interface (JNDI) server is meaningless for a COM client to look up for a remote object. The Web services framework brings a standard of distributed computing infrastructure, in which remote objects are described by WSDL files and can be invoked synchronously/asynchronously by the general acknowledged protocol SOAP regardless of how those objects are developed, e.g. programming language or operation system. In other words, objects are really ready to be shared. The same question arises: how could

these objects be located? A global registry service is needed for all the Web services to be discovered and transacted over the Internet. UDDI is a way to solve this problem.

UDDI is the equivalent of the phone book (yellow page) for locating services providers and Web services. WSDL documents become entries in a UDDI registry where they can be searched and found. The services registered in UDDI are categorized by some standard taxonomy like NAICS (North America Industry Classification System) to help them to be retrieved more effectively. UDDI provides SOAP based interfaces (UDDI APIs) for searching and publishing. The UDDI API consists of three main parts:

- An inquiry API, which is provided for clients who want to locate business and services. It provides methods to perform general searches, as well as specific queries for a known item.
- A publishing API, which enables services to publish information about themselves and to delete entries from the UDDI registry.
- A replication portion of the API, which is really intended for server-to-server communication within the network of UDDI servers.

There are some companies like Microsoft and IBM provide public UDDI for business to register their Web services. [15, 16] At the same time anyone can provide such service using specified tools like Xindice.

# Chapter 3 Infrastructure for Distributed E-Learning

## 3.1 Background

If we look at the history the electronic-based learning system's evolution, it has come a long way. In the early 1960's, the forms of e-learning approaches were just standalone software programs, which are called Computer-Based Instructions (CBI). CBI development focused on automating relatively simple notions of learning and instruction and then developing methods that proved to be effective. [17] Those software programs represented learning theories into different algorithms, and the fast calculation and data recording features of computers were used to determine if those theories are correct. In the later 1960's, groups of researchers began to explore a new approach of e-learning called Intelligent Tutoring Systems (ITS), which was rooted in early artificial intelligence studies like how we learn, master skills and define subject domains. The specifications are distinct from those found in more conventional approaches to CBI. They require ITS to generate instruction in real time and on demand as required by individual learners and support mixed initiative dialogue, allowing free form discussion between the technology and the student or user. [18] Those specifications also led to the initiative of the current distributed learning.

In recent years, major changes and progresses in computing and communications infrastructure especially in the Internet area are converging to produce revolutionary changes in e-learning systems technology. The World Wide Web has essentially reset the development agenda for both CBI and ITS development. Since the technical standards of the Internet turn out to work equally well locally, regionally and globally, the Web has become the universal delivery platform for any kinds of learning materials or contents and it has raised e-learning into a higher stage of distributed learning. Combining traditional Computer-Based Instruction and interactive multimedia technologies with new Web-enabled intelligent tutoring and simulation capabilities are referred to as "Advanced Distributed Learning". [18] Some examples of distributed learning technologies include WWW, email, video conferencing, online chat rooms, newsgroups, discussion forums, whiteboards, simulations and instruction software. A distributed learning system should

be established over a shared and collaborative environment. This environment allows learners or instructors to enter the learning system at any time and from anywhere. Users can get required information and communicate each other through either synchronous or asynchronous way in this environment. The Shared Place is intended to be such an environment.

## 3.2 A Shared Place for Learning

Shared Place [21], which was proposed by CollabWorx [22], supports construction of virtual places on the Web where people can interact and work together. Users of such virtual places can use content and tools that are present at the place for collaborative work. They can bring their own content and share it with others. They can possibly create new content and take it with them to another virtual place or store it in a personal repository for off-line work. From the user's point of view, the Shared Place application is an ordinary Web page with added collaboration functionality. Let's image place for online learners. Such place contains relevant information in HTML or XML form: course contents, simulation programs, course assignments and online examinations. It would also contain collaboration tools such as online chat, audio/video conferencing for synchronous communication, emails, discussion forums for asynchronous communication, shared whiteboards for instant demonstration, and so on.

### 3.2.1 Shared Place Model

The Shared Place Model [21] is shown in Figure 3-1.

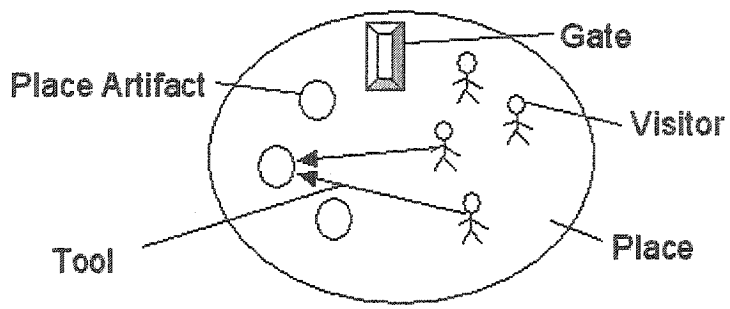



**Figure 3-1** Shared Place Model [21]

The model consists of several basic entities:

- A *Place* corresponds to a physical place in the real world. It provides boundaries where the intended interactions occur. Places can contain other entities like Artifacts, Visitors, Agents, Tools and Gates. Places are persistent and they can interconnect with other Places through Gates. A set of policies like Visitor's authorization and authentication is also defined within the Place. Policies impose various restrictions on the model entities depending on the desired functionality. A well-defined learning system could be a Place.
- A *Visitor* represents the user in a Place. All actions that a user executes on the content of Place are performed by the Visitor. Obeying the roles assigned by the Place, Visitors can enter or leave the Place using Gates, carry their Arifacts and interact with other Visitors using Tools. Learner and instructor are two different types of Visitors of a learning system.
- A *Tool* provides means of interaction between Artifacts and Visitors, as well as between Visitors. Tools can be used to perform various tasks such as: providing awareness information, artifact management, communication or collaborative works. Common Tools in a learning system are: course management programs, learning content management programs, chat rooms, whiteboards and so on. Tools should present different interfaces to the Visitors with different roles.
- An *Artifact* represents content of an arbitrary type that can be created and accessed by a Visitor using Tools. It can be a data container like a Web page, a document, a stream of data, or an image. It also can be a software program developed by Java or C++. Artifacts can be moved from one place to another by Visitors. An Artifact can exit in one of two basic modes – as a Personal or a Place Aritfact.
- An *Agent* may perform actions on Artifacts on behalf of the Visitor while he or she is not present in the Place. Those might be administration tasks like removal of the expired Artifacts, or sending alert emails to Visitors.
- A *Gate* is an entity that enables Visitors to move from one Place to another. It also provides the communication means, by which one Place can interact with another and share Artifacts locate in different Places.

## 3.3 Related Work

### 3.3.1 ARIADNE System & Tools

Since 1997, the Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE) [23] has been involved in standardization activities performed under the auspices of the Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC). [24] The major contribution of ARIADNE is not only the released Metadata recommendation, which is based on the IEEE LTSC Learning Object Metadata (LOM) [25] latest working draft but also the ARIADNE System & Tools that are based on the ARIADNE Knowledge Pool System (KPS). The following figure illustrates the basic architecture of ARIADNE system.

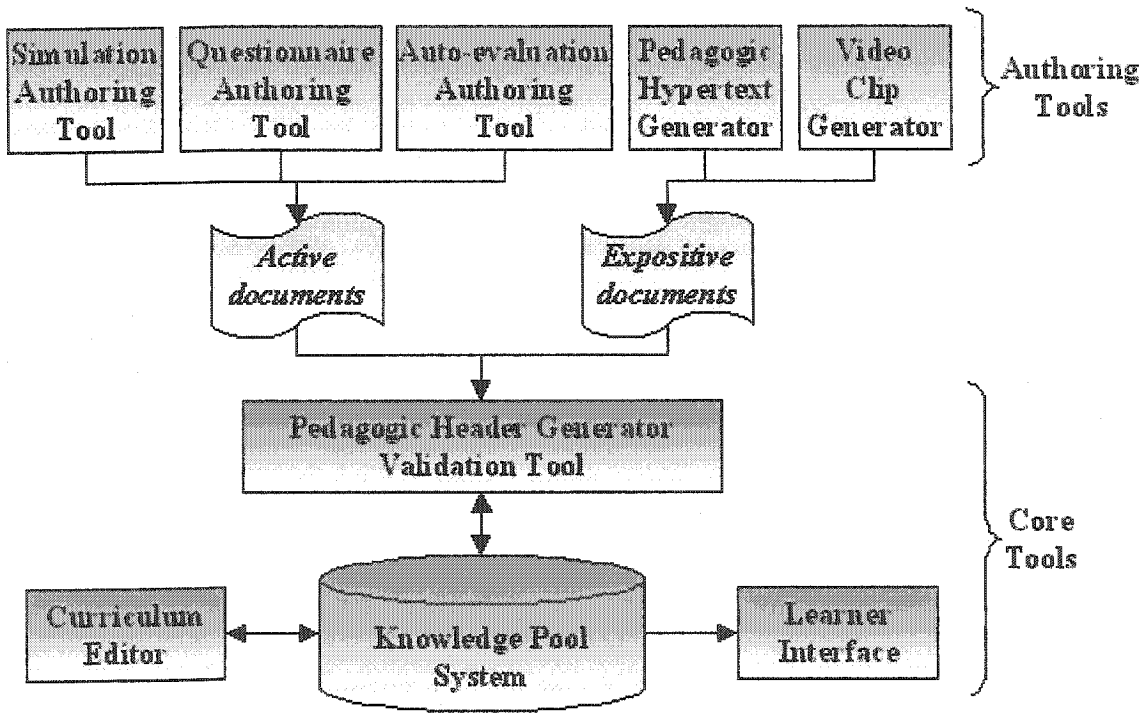



**Figure 3-2** The Architecture of ARIADNE System [23]

The whole system consists of three kinds of tools: Core Tools, Authoring Tools and Communication Tools. The KPS, which is the core of the Core Tools, is a distributed database system in which stores the most basic learning material units called learning objects that are described by the ARIADNE metadata. There are several replications of KPS repositories locating almost all over Europe. Besides an indexation tool allowing for querying, creating, updating, validating and storing metadata as well as the learning

object in to KPS, the Core Tools also includes a curriculum editor for defining online courses and an interface tool for learner accessing. The Authoring Tools are used to create instructional simulations, present questionnaires, generate pedagogic learning resources and create auto-evaluation exercises. The Communication Tools provide many synchronous or asynchronous communication tools that permit a structured collaboration among distance users.

Even though the ARIADNE system is still being improved in some aspects like migrating to Web-based architecture and approaching platform independent, it is still a good working implementation example of the Shared Place for learning.

### 3.3.2 SCORM

The Sharable Content Object Reference Model (SCORM) is defined by the Advanced Distributed Learning (ADL) Initiative [26], sponsored by the Office of the Secretary of Defence (OSD) in United States. SCORM is a model that references a set of interrelated technical specifications and guidelines designed to meet Department of Defence's (DoD) high-level requirements for Web-based learning content. [18]

Building upon the Internet standards and the Shared Place infrastructures system developers have switched their focuses more on how to define reusable learning objects and develop new content models. A key ADL requirement for learning content is the ability to reuse instructional components in multiple applications of the tools used to create them. [18] SCORM is intended to be such a model that deals with the reusability, accessibility and interoperability of sharable learning objects. SCORM applies several technology standards: IMS Global Learning Consortium, Inc. [27], the Aviation Industry CBT (Computer-Based Training) Committee (AICC) [28], the Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE) [23], and the IEEE Learning Technology Standards Committee (LTSC). [24]

SCORM is divided into two main parts: the Content Aggregation Model and the Run-Time Environment.

#### 3.3.2.1 Content Aggregation Model

The SCORM Content Aggregation Model represents a pedagogically neutral means for designers and implementers of instruction to aggregate learning resources for the purpose

of delivering a desired learning experience. [19] It involves the creation, discovery and gathering together, or aggregation, of simple assets into more complex learning resources and then organizing the resources into a predefined sequence of delivery.

The SCORM Content Model is made up of the following components: Assets, Sharable Content Objects (SCO) and Content Aggregations. Asset is the most basic form of a learning object. It is an electronic representation of media, text, image, sound or web page. A SCO represents a collection of one or more Assets that include a specific launchable asset that utilizes the Run-Time Environment to communicate other learning systems. A Content Aggregation is a map (content structure) that can be used to aggregate learning resources into a cohesive unit of instruction (e.g. course, chapter, module, etc.). The IEEE LTSC Learning Object Metadata (LOM) [25] is applied to Assets, SCOs and Content Aggregations to describe them in a consistent fashion such that they can be searched for and discovered within and across systems to further facilitating and reuse. Figure 3-2 below shows an example of the Content Aggregation Model. [19]

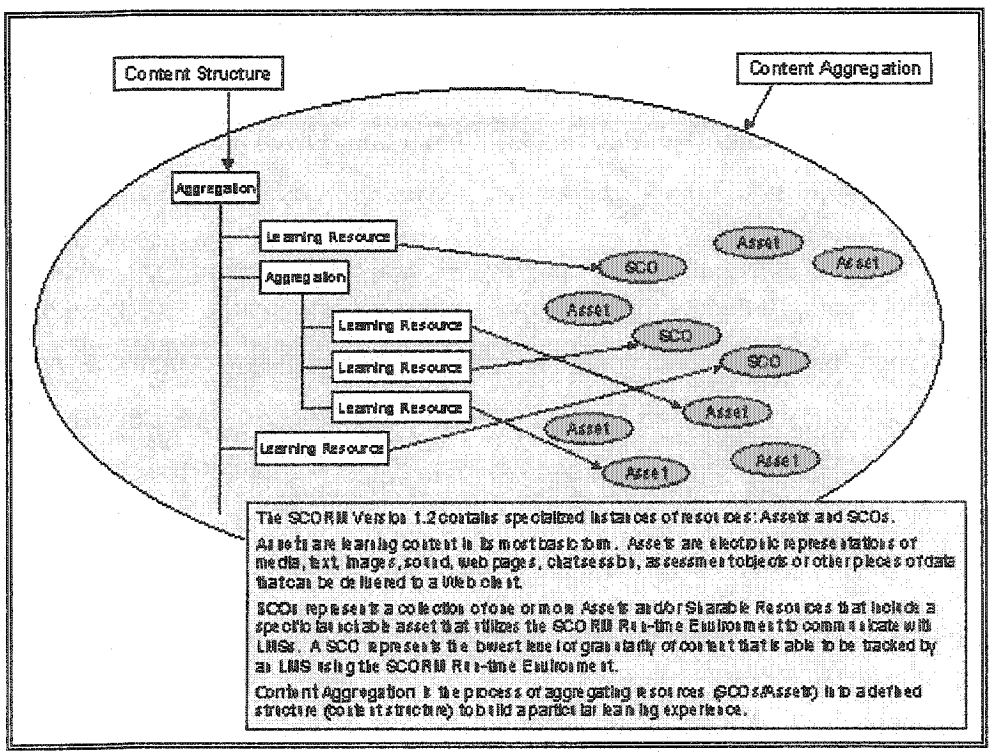



**Figure 3-3** The SCORM Content Aggregation Model [19]

### 3.3.2.2 Run-Time Environment

The SCORM Run-Time Environment provides a means for interoperability between Sharable Content Object-based learning content and Learning Management Systems. There are three aspects of the Run-Time Environment are Launch, Application Program Interface (API) and Data Model. [20]

The Launch mechanism defines a common way for Learning Management Systems (LMS) to start Web-based learning resources. This mechanism defines the procedures and responsibilities for the establishment of communication between the delivered learning resource and the LMS. The API is the communication mechanism or informing the LMS of the state of the learning resource (e.g., initialized, finished or got an error), and is used for getting and setting data (e.g., score, time, etc.) between the LMS and the SCOs. A Data Model is a standard set of data elements used to define the information being communicated, such as, the status of the learning resource. The figure below shows these three aspects of the Run-Time Environment. [20]

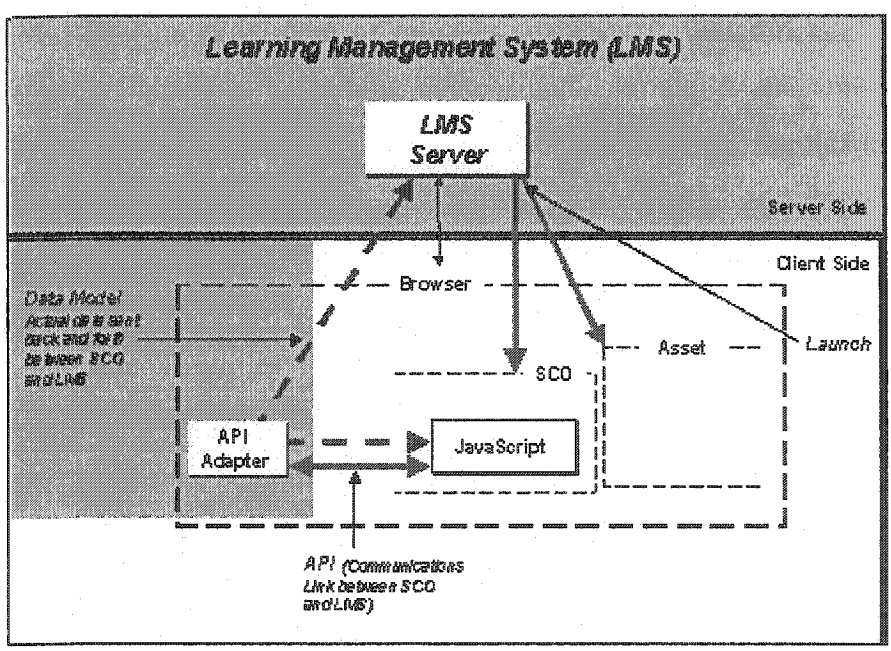



**Figure 3-4** The SCORM Run-Time Environment [20]

## 3.4 Distributed Learning Future

The Shared Place Model has provided a birds-eye-view of what kind of virtual environment that a distributed learning system should reside in. And some significant

researches like ARIADNE and SCORM were already launched for pursuing the high requirements of accessibility, interoperability and reusability. However, those approaches still need further improvements and consummation. For example, not all ARIADNE tools are Web-based yet, which means users can access these services only by systems that are equipped with the required software programs. SCORM provides systematic mechanisms for learning content's aggregation and delivery, but it lacks of the mechanism of discovering, locating and publishing in order to save system developers' time spent on creating same learning objects or services. And most learning systems do not support the compatibility with other objects or services provided by the third party vendors due to the limitations of the traditional distributed computing technologies. Therefore, there is a demand of a new framework that can provide a true flexible model with open standards in which all the learning objects and services can be easily shared and loosely connected.

The general view of next generation of distributed learning environment should be like this: all the sharable learning objects and public learning management services are described by standard metadata and binding information published in a universal repository for discovery and locating. Any independent learning system can embed desired learning objects found from this repository to its learning resources as long as the metadata of the objects matches the learning resource aggregator's expectation. Learning systems even deployed in totally different platforms can still interact with each other through the services they posted over some standard communication protocols they already complied. This framework leverages most learning systems to approach the Shared Place Model.

# Chapter 4 Building a Web Services Oriented E-Learning System

This chapter provides a general overview and concepts of a Web services oriented distributed e-learning framework that fulfills the high-level system requirements listed in the first section. Subsequently, the later sections describe the analysis and design details of an e-learning system based on this framework.

## 4.1 High-Level System Requirement

In order to raise the traditional Web-based distributed learning to a new high level achieving a true dynamic model the following requirements must be given special consideration.

*Open architecture and interfaces:* The framework should have an open architecture and open application interfaces to enable interaction and integration seamlessly between educational institutions, learning service providers, and other entities that enable distributed learning. The architecture is able to take advantage of the open, dynamic nature of the Web by supporting just-in-time application integration.

*High interoperability for information exchange:* The framework should have the ability to take learning components or applications developed in one location with one set of tools or platform and use them in another location with a different set of tools or platform. In other words, these components can be retrieved from anywhere, anytime with any device (wired or wireless).

*Flexibility:* All the sharable learning objects in this model are loosely coupled, which means the system should offer an dynamic mechanism that allow developers to easily add or remove components at any time.

*Accessibility:* The learning objects can be published with well-defined description in a universal repository for search, discovery and retrieval by other remote applications that need them.

*Durability and Reusability:* The system based on this framework should be designed in an object-oriented model, in which the learning materials or applications are treated as encapsulated components with well defined interfaces. As long as the

interfaces remain consistent, these components can withstand technology changes without redesign, reconfiguration or recoding and can be reused in multiple applications and contexts.

*Compatibility with other systems:* The system based on this framework should provide an open standard interface to communicate with the third party learning applications.

## 4.2 Proposed Framework

Web services enable most of those key requirements listed in the previous section for distributed e-learning systems. Based on open standards like XML and SOAP, they define a means by which learning services can be published, discovered and invoked. They support directories of organization and services where entries are added, modified or deleted dynamically, thereby delivering the most current search and identification results to learning service requestor. They also define a standard way to describe and invoke a Web service interface so that the learning application can easily connect to it.

In this section a framework is to be designed and modeled that is able to comply with most of the requirements outlined in the previous chapter. A block diagram of Web services oriented framework with multiple tiers for distributed e-learning system is illustrated in Figure 4-1:

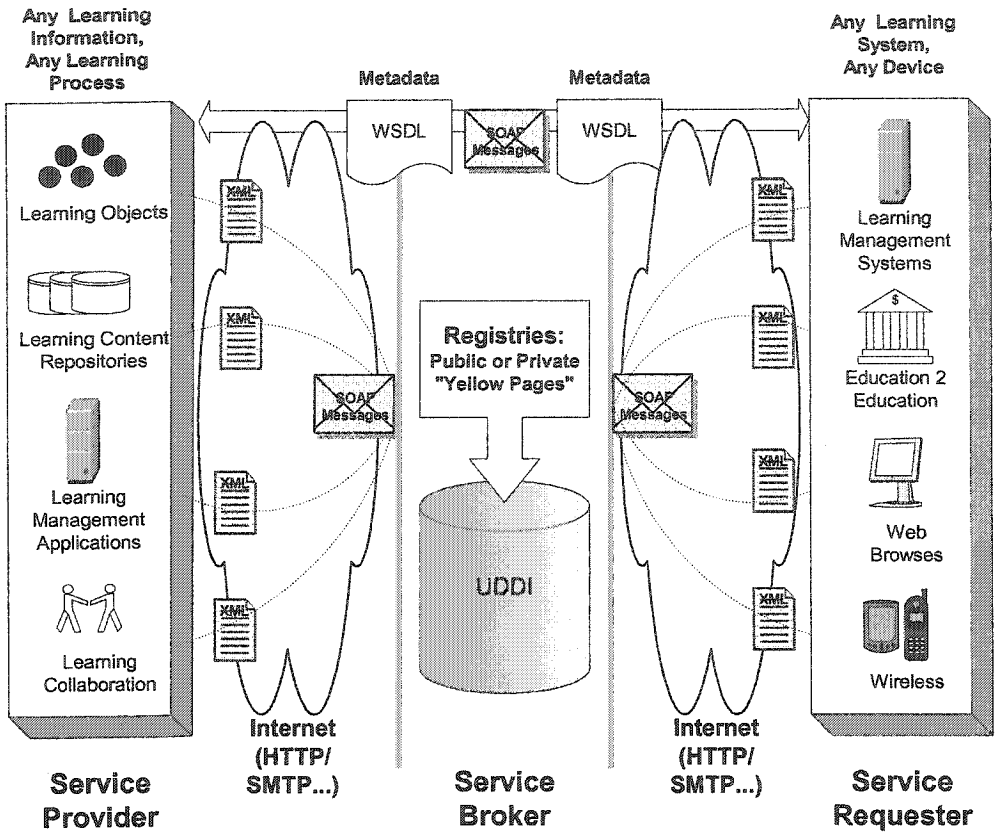



**Figure 4-1** A Web Services Oriented Framework with Multiple Tiers for Distributed E-Learning System

In this figure we have different components. All the learning objects, contents and applications are built as services with XML front end and described with some generally available terms such as metadata so that service requesters are able to search and map these services to their needs. IEEE Learning Technology Standards Committee (IEEE LTSC) [24] and IMS Global Learning Consortium [27] have proposed some learning object metadata standards for reference. A *Service provider* provides any learning information, material or process as a self-contained, self-describing modular service. It could contain one or more of the following:

- Learning Objects and Learning Contents are reusable learning resources as defined by SCORM Content Model [26]. Learning object can be any object with clear invoking interface defined like a Flash object, a JavaScript function, an animation program, etc. Learning content can be any content stored in a database repository or a file system like a HTML page, an XML document, or a JPEG image, etc.

- Learning Management Applications can be any program or function considered as shareable services by businesses or educational institutions, such as course administration services, user management services, testing/assessment services, etc.
- Learning Collaboration includes both synchronous and asynchronous communications among different parties. Synchronous communication could be instant messaging/chat, shared whiteboard, or teleconferencing, etc. Asynchronous communication could be email, discussion forum, or news group, etc.

A *Service broker* provides a universal Web services registry (UDDI), in which service provider and service requester can publish learning services and find desired learning services respectively. Once the learning objects, contents and applications are built as services, they are described by a WSDL document to let requesters know how to invoke them. Then the WSDL file and XML metadata of the services are wrapped in a SOAP message that will be sent to the UDDI. All the necessary information for discovery is registered in the UDDI with directory and key word supports just like a "Yellow Page". Through this "Yellow Page", learning service requesters are able to find learning service providers, learning objects services, learning contents services or learning service applications by sending an SOAP request over HTTP. In fact, UDDI facilitates the loose connection between any two or more applications. Since services registered in UDDI are located by a Universally Unique ID (UUID), once the service provider wants to move a service from one server to another, the only thing to do is to update the binding information of the service like its URL (Uniform Resource Locater) from UDDI and the serve requester always gets the up to date information without changing anything.

A *Service requester* is the consisting of any application that requests a particular service. Based on the binding information of the services found in UDDI, request applications can directly contact the services regardless of what kind platform used in each side since SOAP is the standard communication protocol. A Service requester could be a learning management system, which needs some specific services for integration such as learning contents, remote course registration, shared whiteboard, etc.

So, once a service provider creates a learning object like a Java function that is described by a standard WSDL file or a learning content with a URL specified, they both should have application-neutral XML interfaces and should be described by the standard

learning metadata like LOM. Those learning resources will be published to a UDDI through a standard delivery mechanism and also a searching mechanism is needed here for services requesters retrieving. Since SOAP is the communication protocol among them, the whole framework is highly interoperable, which means the learning objects or contents registered in the UDDI are available for any application residing on any platform. However, the new transforming mechanism over SOAP should not slow down the system performance of those learning resources' publishing, retrieving and invoking. In other words, the system performance has to be acceptable or as good as those with traditional distributed mechanisms.

The framework enables a true flexible integration among educational institutions or businesses, e.g. the single sing-on process for a student to register a course from one institution to another. A system implemented based on this framework would be able to support of diverse and heterogeneous end systems, such as Desktop PC, PDA (Personal Digital Assistant), cell phone, etc. (the details of wireless access Web services are described in Chapter 5)

## 4.3 Typical LMS Interaction Scenarios

Figure 4-2 illustrates that how two Learning Management Systems (LMS) interact with each other to share their services via the framework.

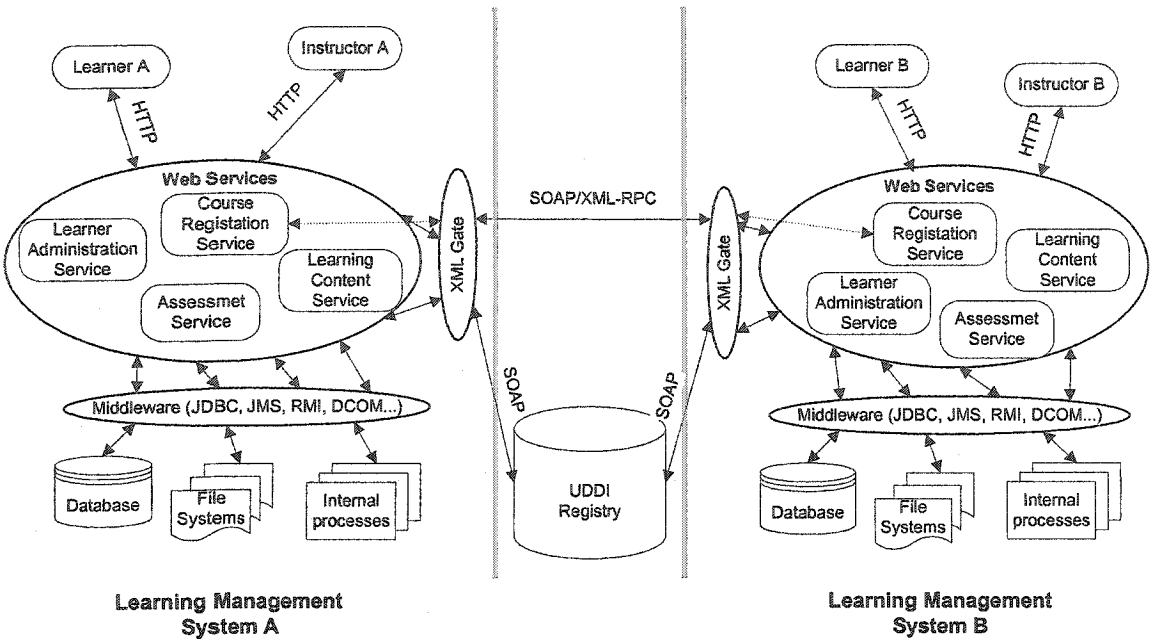



**Figure 4-2** Two Learning Management Systems Interactions via the Framework

LMS A and LMS B are two independent Web based learning management systems with their own system infrastructure. Learners or instructors can access the backend database, file systems and other internal processes through any kind of middleware already built in their systems such as JDBC (Java Data Base Connection), JMS (Java Message Service), or RMI (Remote Method Invocation), etc. Once the sharable learning objects, learning services such as course registration, learner administration, assessment services and collaboration tools are built as standard Web services components and published in the UDDI registry, two independent systems can interact with each other on the application-to-application level by the SOAP messaging or XML based RPC. The XML Gate is the key handler for dealing with standard SOAP request & response of the message driven communication mechanism. The XML Gate ensures the flexibility and interoperability of this framework in which all the service components are connected only by the standard, platform independent XML glues.

- Scenario 1: Single Sign-On Process

  Learner A wants to register courses in both LMS A and LMS B. It is unnecessary for her to log on the two systems separately to retrieve courses information, register courses and maintain registration information cause all those actions can be done through one portal by taking advantage of the interactions between the two systems'

registration services. Learner A can register a course through the course registration service of LMS A (in educational institution A) by providing the user information like user id, password and related course information like course id. Meanwhile, Learner A can also get courses information and register courses offered by educational institution B by her same authentication information within the same user interface because her user information has been wrapped into a SOAP message and sent to the URL of LMS B's course registration service that already had been found from UDDI. Then the corresponding course information is sent back and shown to learner A after she has been authorized. With the communication at the application level, learners or instructors do not have to logging off from one system and singing on to another to do the same actions. This process is called single sign-on, which really provides a transparent and integrated learning environment portal for all users.

- Scenario 2: Learning Content Aggregation

  LMS A and LMS B are SCORM enabled learning systems. They have their own learning objects – Sharable Content Objects (SCO) reside in their databases, file systems or any other kind repositories. Once an instructor wants to develop and deliver a new learning resource (e.g. course, chapter or module), he just aggregates all those reusable, shareable and interoperable SCOs using the SCORM Aggregation Model. Now based on the open structure of Web services framework, SCOs are available not only at your local repositories but also over the World Wide Web. For example, if the instructor cannot find an expected learning object in the repositories of LMS A, he can just search it from the UDDI using standard metadata instead of creating a new one to the system. Once the result is matched, he just simply takes the resource URL to embed in his content (if the learning object is a HTML or an image) or communicates with the object through SOAP messaging or XML-RPC in his content aggregation scripts (if the learning object is an algorithm function can be invoked). This process saves the instructor a lot of time in building the learning resource by without recreating same objects that already exist somewhere else over the network.

## 4.4 System Analysis

We used the industry-standard language: Unified Modeling Language (UML) to specify, visualize and construct the artefacts of the system. The goal of the analysis phase is to truly understand the requirements for the new system and develop a system concept that addresses them. [29] The requirements based on the previous scenarios are first visualized into a use case model, then a structural model is derived to realize those use cases, and finally a behavioural model is constructed to specify the internal processes.

### 4.4.1 Use Case Model

Use case model consists of actors, use cases and relationships. There are four actors in our use case model: Learner, Instructor, Administrator and External Learning Systems. The external learning systems represent the service requesters that can take advantage of Web services technologies to get the shared learning services provided by the system (service provider) over the Internet. The following table gives the detailed use cases related to these actors.

| Actors | Use Cases |
|---|---|
| Learner | • Browse course information<br>• Register courses<br>• Take online lessons, tutorials or assessments<br>• Collaborate with other learners or instructors using discussion forum, on line chat or whiteboard |
| Instructor | • Construct online learning resources like online courses, tutorials and assessments<br>• Create learning objects<br>• Search learning objects<br>• Evaluate learner performance<br>• Collaborate with other instructors or learners using discussion forum, on line chat or whiteboard |
| Administrator | • Maintain course related information like course time schedule, course registrations, etc<br>• Maintain user personal information<br>• Manage user privileges ad preferences |

| | |
|---|---|
| | • Publish/Remove learning services to UDDI |
| External Learning Systems[2] | • Search learning services from UDDI<br>• Get shared course information<br>• Register shared courses remotely<br>• Get or invoke shared learning objects remotely |

**Table 4-1** Actors and Use Cases

The follows are the use case diagrams, which illustrate the relationships between actors and use cases more visually.

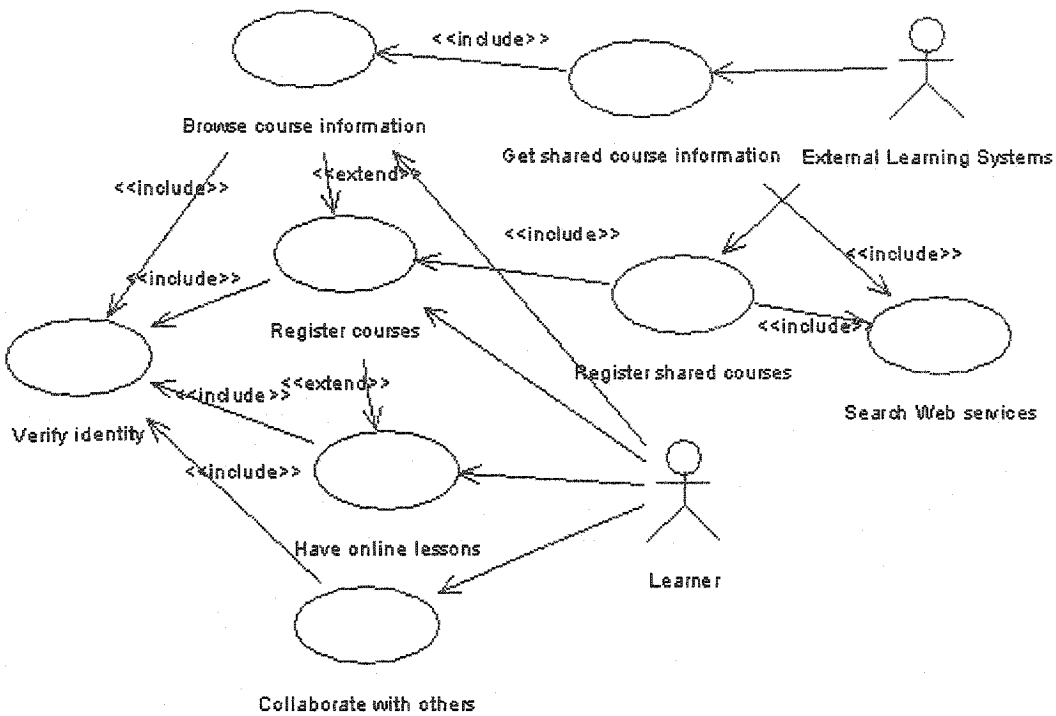



**Figure 4-3** Use Case Diagram for Actors: Learner and External Learning Systems

[2] The following sections of analysis and design focus more on how to realize the uses cases of this actor because this actor is the main benefitor of the Web services framework.

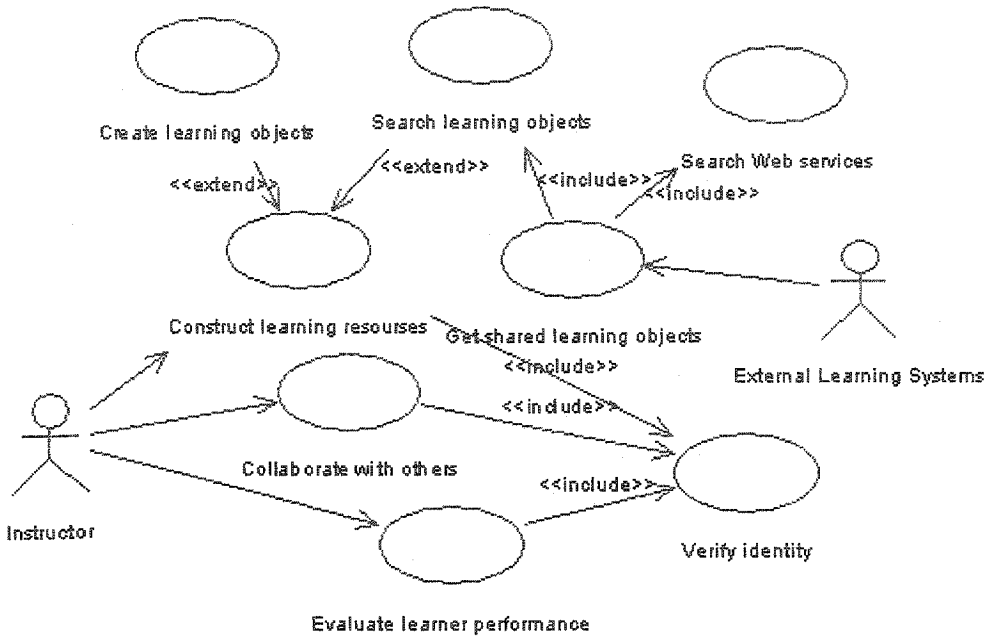



**Figure 4-4** Use Case Diagram for Actors: Instructor and External Learning Systems

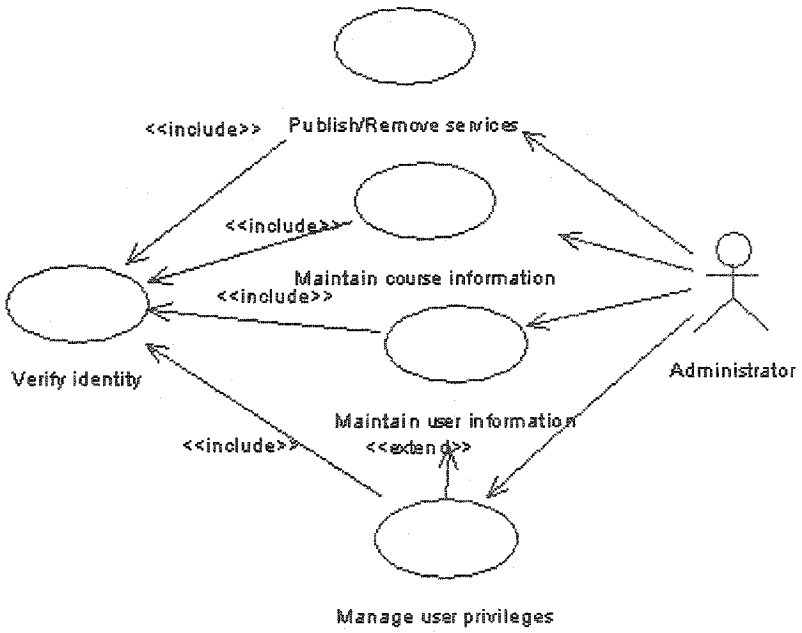



**Figure 4-5** Use Case Diagram for Actor: Administrator

Some association relationships like "extends" and "includes" are used in our model. An "extends" relationship between use cases means one use case optionally adds the functionality of the other use case when certain conditions are met. An "includes"

relationship means one use case completely encompasses all the functionality of another use case. [30]

### 4.4.2 Structural Model

A structural model or a conceptual model describing the realization of use cases serves as an abstraction of the design model, which is depicted in section 4.5. The goal of a structural model is to create a preliminary mapping of required behaviour onto modeling elements in the system. [30] The most common modeling elements in the structural model are called analysis classes[3]. There are three types of analysis classes throughout the structural model.

- Boundary – A boundary class is a stereotyped class that models the interaction between one or more actors and the system. It can be a window, a web form, or a printer interface, etc.
- Control – A control class models behaviour specific to one or a few use cases. Control classes coordinate system behaviour and they handle the main tasks and control flows.
- Entity – An entity class models information stored by the system and its associated behaviour.

Before the structural model is introduced, its super ordination: use case realizations should be demonstrated first. Here is the use case realization diagram[4]:

---

[3] Analysis classes are stereotyped classes. Stereotype is a meta-classification of an element, for example actor and use case are stereotypes for elements in use case diagram, while boundary, control and entity are stereotypes for elements in structural diagram.

[4] This diagram realizes part of the use cases that are associated with actor: external learning systems like get shared course information and register shared course. Those realizations implemented in the current running system called UbiLearn that designed and developed at Multimedia Communication Research Lab at the University of Ottawa are good enough to show the feasibility of the proposed Web service framework. Other use cases will be realized in the later version of UbiLearn.

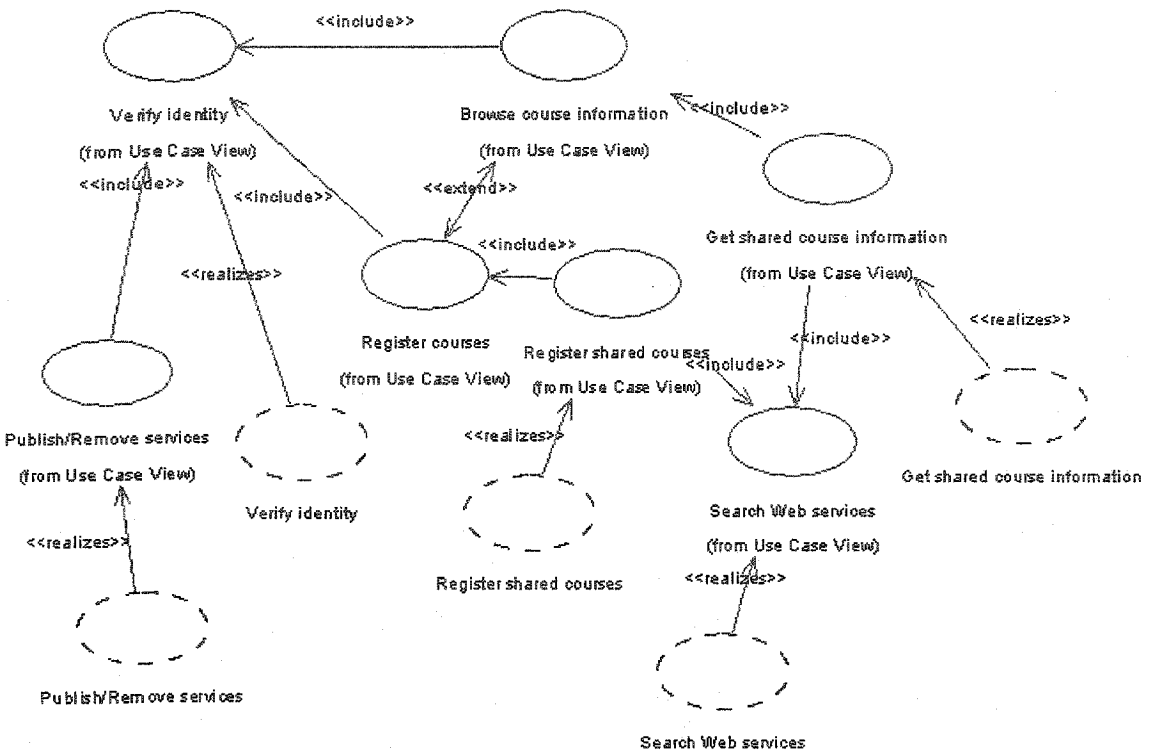



**Figure 4-6** Use Case Realization Diagram

As we can see from the diagram, parts of the use cases associated with actors of external learning systems and administrator have been realized, which are specified by the dotted circles. Let's take a look at the detailed structure of each realization.

- The realization of "Verify identity":

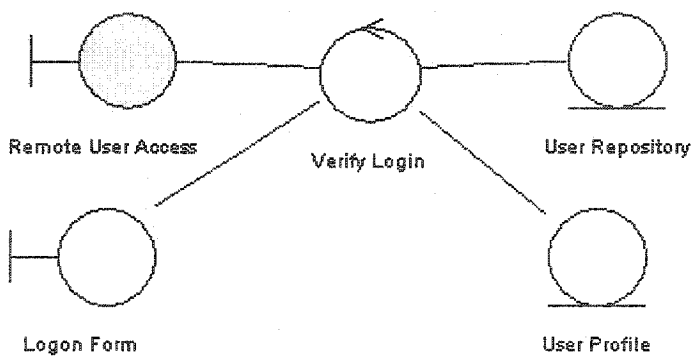



**Figure 4-7** Structural Diagram for the Realization of Use Case: Verify identity[5]

[5] 1) ⊢◯ - Boundary class; ◯ - Control class; ◯ - Entity class 2) The analysis classes in shadow are related to Web services classes

A learner or an instructor can log on to the system through the logon web form. The system verifies the user identification information with the user repository then stores the current user information to the user profile session. For external learning systems to log in the system, only a remote service interface is needed.

- The realization of "Get shared course information":

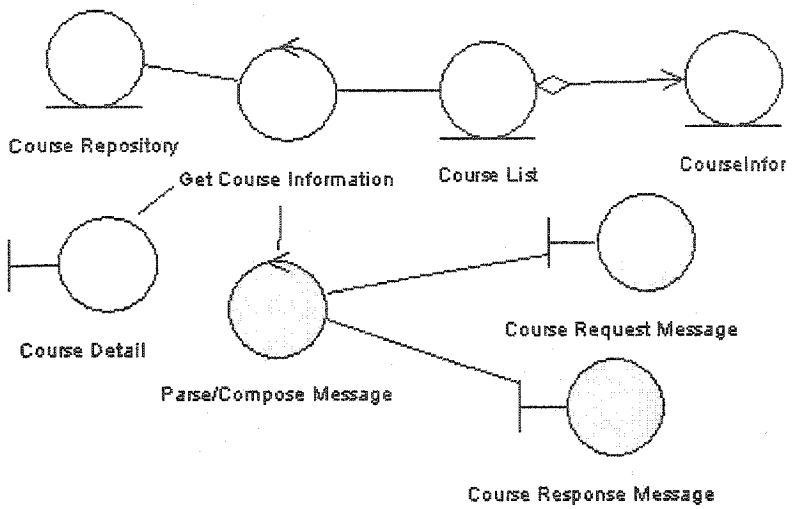



**Figure 4-8** Structural Diagram for the Realization of Use Case: Get Shared Course Information

A learner can browse course detailed information through the course detail web page. The system retrieves the course information for the repository and stores into a course list object. If external learning systems can share those courses, they just send course request message to a "Parse/Compose Message" control class that interprets the messages and communicates with the internal process, and then get the response of course information.

- The realization of "Register shared courses":

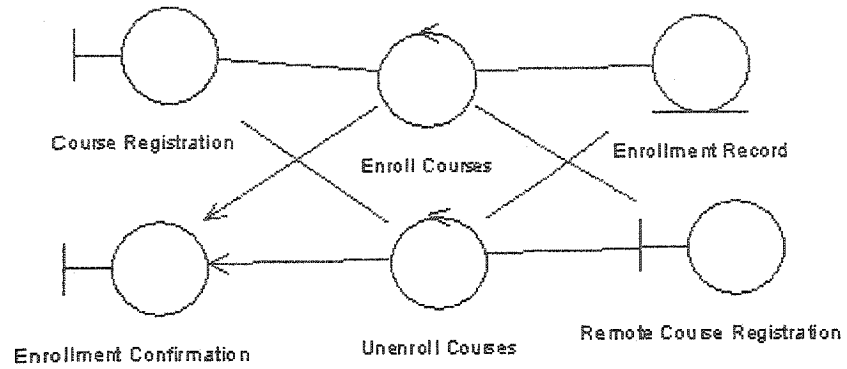



**Figure 4-9** Structural Diagram for the Realization of Use Case: Register Shared Courses

A learner can register a course through the course registration form, in which system calls "Enrol" or "Unenrol" process to update the learner's enrolment record and returns the confirmation. A remote course registration service interface is needed here for external learning systems to invoke "Enrol" or "Unenrol" process remotely.

- The realization of "Search Web services":

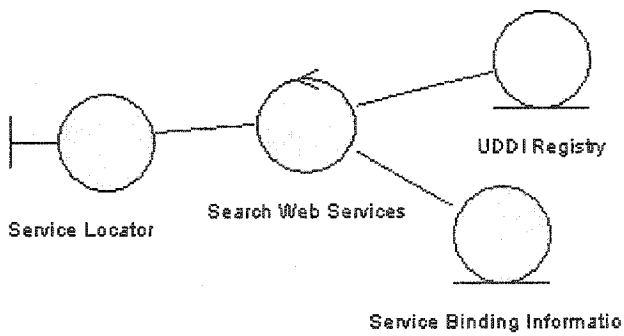



**Figure 4-10** Structural Diagram for the Realization of Use Case: Search Web Services

Before external learning systems invoke the sharable learning services, they need a mechanism to search, locate and understand those services. So they should be able to search Web services from UDDI registry through a service locator interface and get all the service binding information.

- The realization of "Publish/Remove Web services":

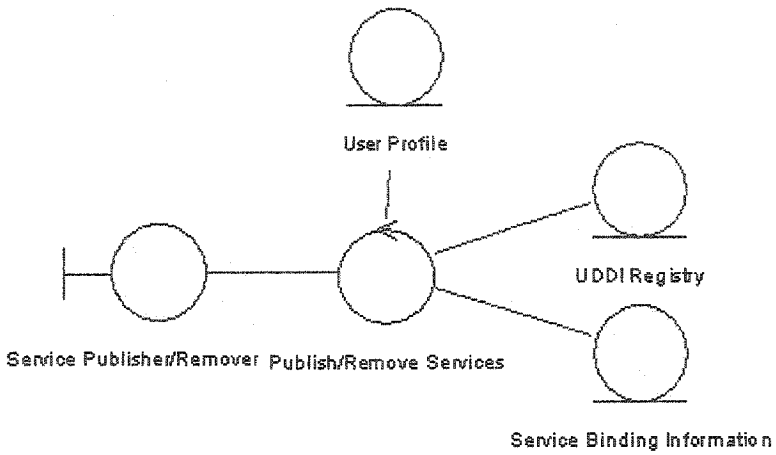



**Figure 4-11** Structural Diagram for the Realization of Use Case: Publish/Remove Web Services

A system administrator can publish a new learning service to a UDDI registry and remove a learning service from a UDDI registry through the service publisher/remover interface.

Not all the use case realizations are described in detail above cause some of them are supposed to be modeled in the business logic tier, which is not the main point of this thesis. What we concern most is how to design and implement the Web service tier within the whole architecture. (see Section 2.2.2)

### 4.4.3 Behavioural Model

Structural model provides a static view of what analysis classes should be involved and interacted to model a specific use case realization. Behavioural model, which is based on structural model, contains two more elements like *sequence* and *message* to demonstrate a more dynamic aspect of realizations. Behavioural model is usually represented by sequence diagrams or collaboration diagrams. Collaboration diagrams are equivalent to sequence diagrams, but they emphasize the flow of messages through a set of objects, while the sequence diagrams focus on the time ordering of the messages being passed. [29] Sequence diagrams are used in our behaviour model. The same use case realizations are illustrated by sequence diagrams shown below.

- The realization of "Verify identity":

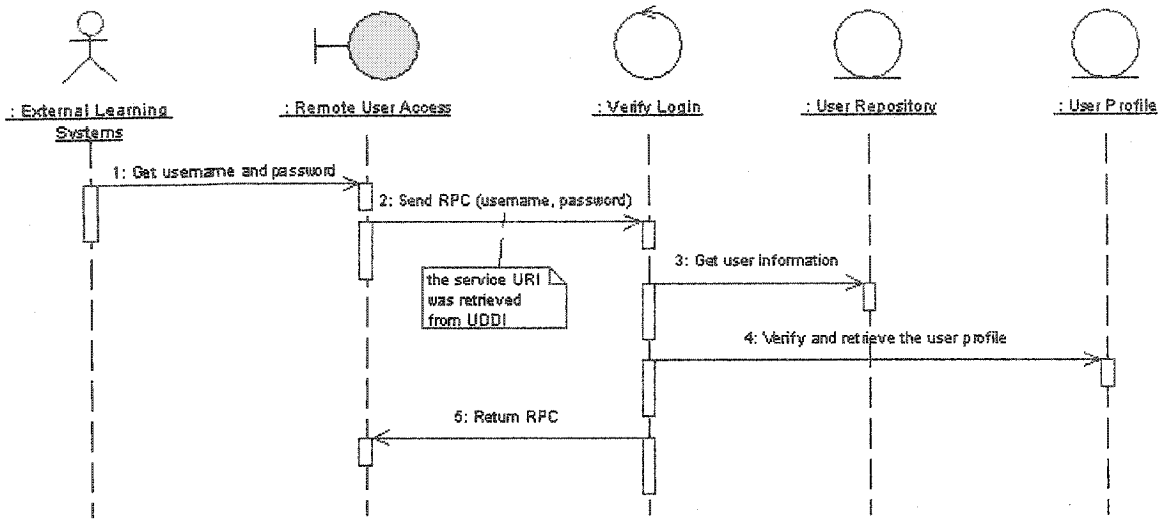



**Figure 4-12** Sequence Diagram for the Realization of Use Case: Verify identity

Figure 4-12 clearly shows the sequential flows and transmission messages of how an external learning system logs into a remote system. The remote access interface is a XML based RPC client.

- The realization of "Get shared course information":

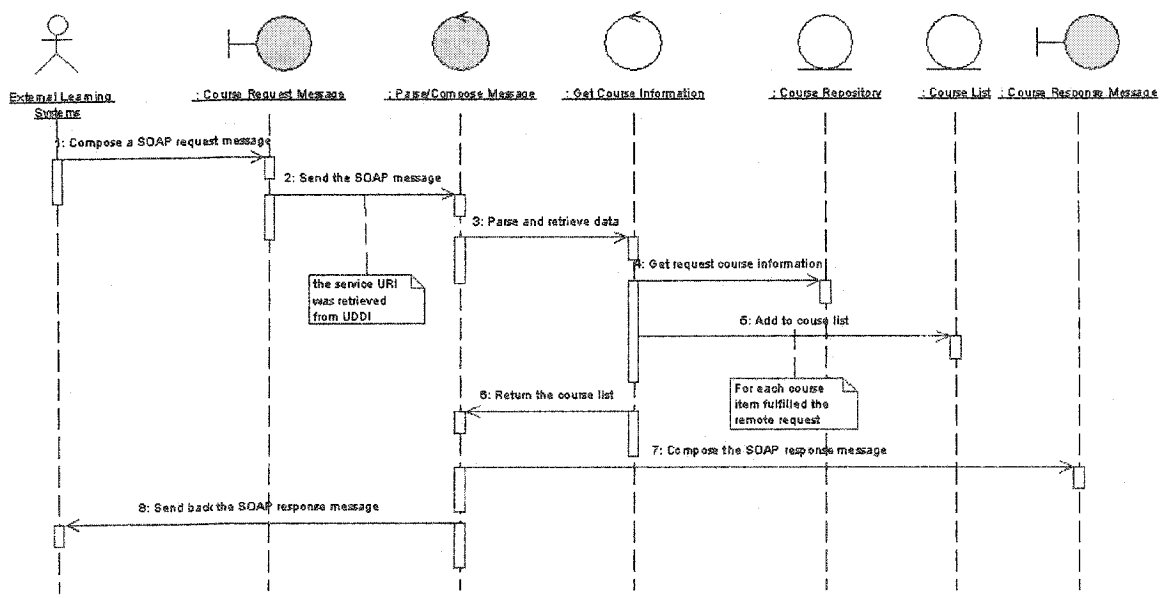



**Figure 4-13** Sequence Diagram for the Realization of Use Case: Get Shared Course Information

Figure 4-13 shows the sequential details of how an external learning system get shared course information. Both the course request and course information response are transmitted by SOAP.

- The realization of "Register shared courses":

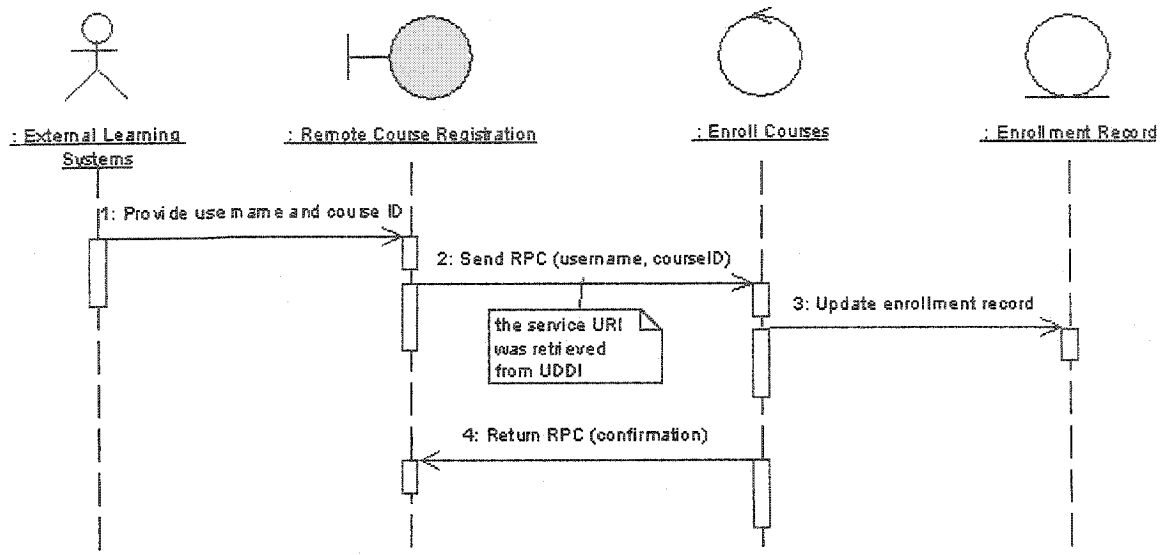



**Figure 4-14** Sequence Diagram for the Realization of Use Case: Register Shared Courses

The course registration for an external learning system can be done by a simple XML-RPC invocation with its username and the course ID.

- The realization of "Search Web services":

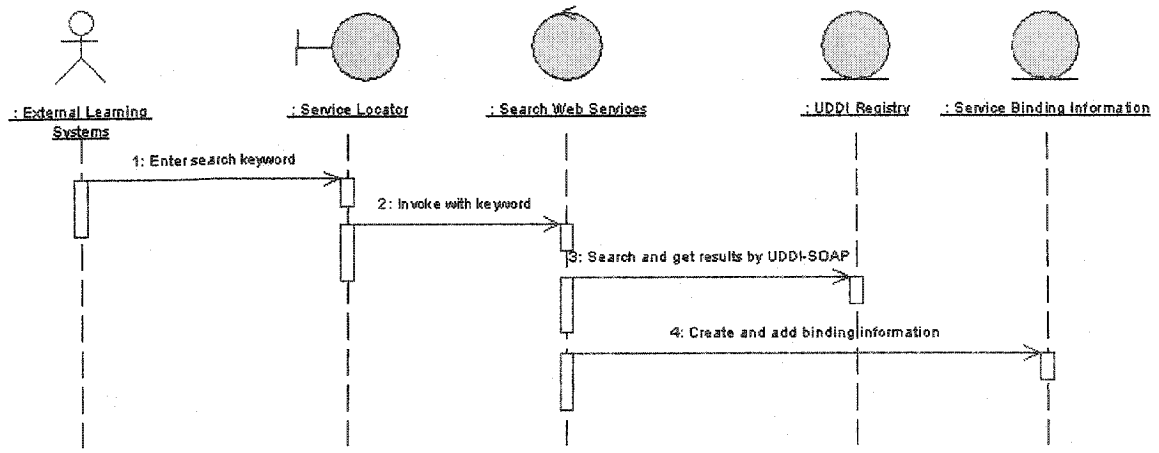



**Figure 4-15** Sequence Diagram for the Realization of Use Case: Search Web Services

Figure 4-15 shows how an external learning system uses the service locator interface to search its desired learning services and get their binding information through UDDI-SOAP messages.

- The realization of "Publish/Remove Web services":

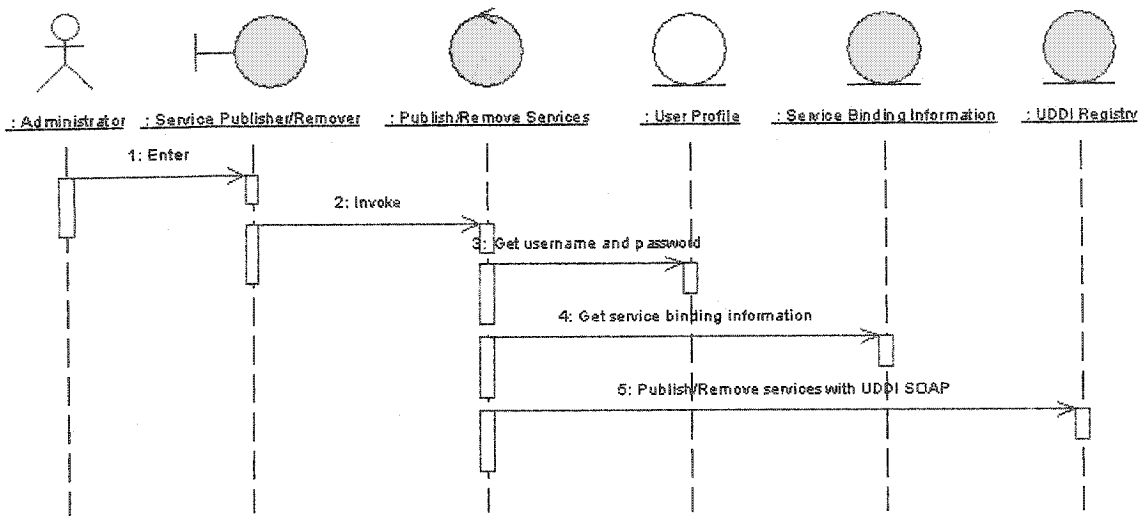



**Figure 4-16** Sequence Diagram for the Realization of Use Case: Publish/Remove Web Services

A system administrator also uses UDDI SOAP to publish and remove Web services. He needs his user identity to be verified before he can manipulate UDDI registry.

## 4.5 System Design

The system design models refine the system analysis models by adding system environment details to them and refining the problem domain information already contained in the analysis models. [29] This section encompasses three major issues of the system design[6]: the overall system architecture, the design pattern and the class design.

### 4.5.1 Overall System Architecture

A general Web services architecture is described in section 2.2.2. The architecture in this section is more focusing on the flows of how two independent learning systems interact via Web services.

[6] The system design also includes the database design and the user interface design, which are not explained here but in the appendix because they are less relevant to the Web services components.

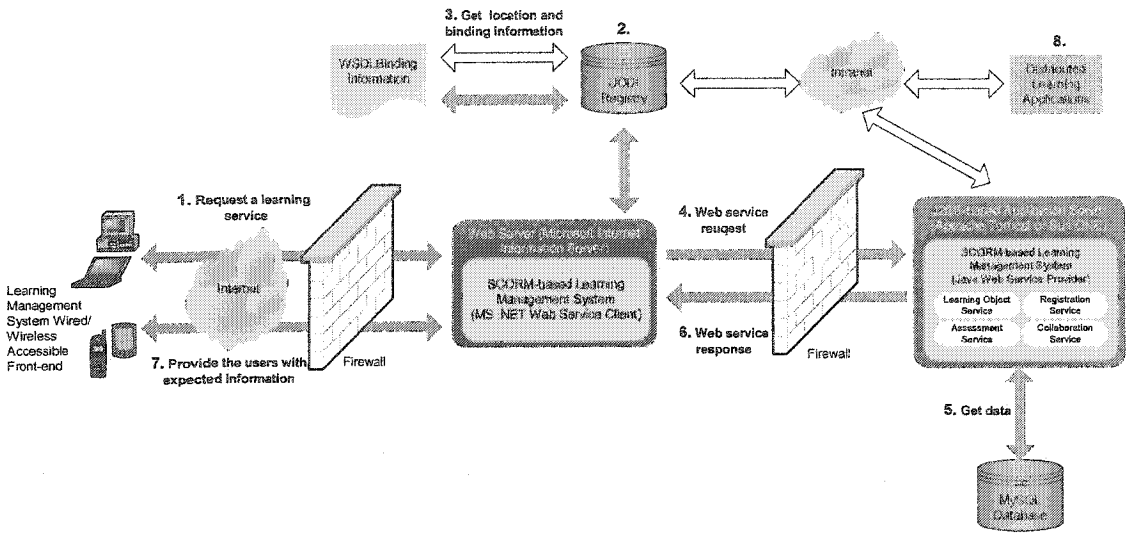



**Figure 4-17** System Architecture for Two Learning Systems Interaction via Web Services

Some vendor names and products mentioned in the architecture are used to demonstrate the interoperability of learning services between different running platforms. The sequence of steps is as follows:

1. A learner or an instructor requests a learning service (e.g. registration of a remote course) from a Web page or a wireless device front-end. The request is passed over the local learning management system running on a Microsoft IIS (Internet Information Server). Here it is assumed that the user has already successfully logged in.
2. The .NET[7]-based [39] LMS gets the information about the registration service made available by J2EE[8] (Java 2 Enterprise Edition) [38] application server by doing a look up in a UDDI registry.
3. The location of and WSDL binding information for the registration service is sent to the service requester as a SOAP-based message.
4. The service requester invokes the service published by the service provider, passing the username and course ID as a part of a SOAP-based message.

---

[7] Microsoft .NET is an XML Web services platform that will enable developers to create programs that transcend device boundaries and fully harness the connectivity of the Internet.

[8] Java 2 Enterprise Edition (J2EE) manages the infrastructure and supports the Web services to enable development of secure, robust and interoperable business applications.

5. The actual implementation of the service is provided by the objects running within another J2EE based application server. The objects use the JDBC (Java Data Base Connection) API to get information from the database.
6. The objects send the Web service response (registration success or fail) to the service requester application as a SOAP-based message.
7. The response is formatted in a browser or wireless device recognizable format and sent back to the client application.
8. Other distributed learning applications such as service publication can be invoked through some middleware within the intranet.

### 4.5.2 Design Pattern: Model-View-Control

Model-View-Control (MVC) is a recommended architecture design pattern for interactive applications. MVC organizes an interactive application into three separate models: one for the application model wit its data representation and business logic, the second for views that provide data presentation and user input, and the third for a controller to dispatch requests and control flow. [32] The architecture of MVC design pattern can be illustrated as follow:

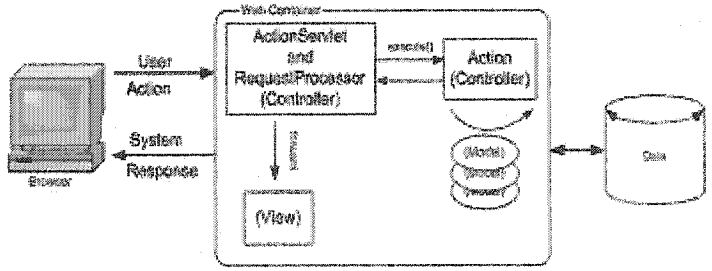



**Figure 4-18** Architecture of Model-View-Control Design Pattern

When the Web container receives a user action such as an incoming HTTP request, it invokes an active element "ActionServelt" to interpret the request and dispatch to the business logic operation in the application model. Based on the results of the operation and state of the model, the controller like "RequestProcessor" forwards the results to the next view to display. Finally, the controller generates the selected view and transmits it to the client browser. We can see that MVC separates the data presentation and business logic into different layers so that the web applications are more extensible, maintainable

and modularized, which encourages the applications to share those operations as Web services.

Struts, which is developed by the Apache Jakarta Project [33], provides an open source framework based on MVC design pattern. The Struts framework is a flexible control layer with some standard Java technologies like Java Servlets, JavaBeans, ResourceBundles, and XML. Struts is used to build our Web based Learning Management Systems throughout the project and it helps our design and development follow a professional and systematic mechanism.

### 4.5.3 Class Design

Throughout the analysis phase we have discussed, we see that there are four major requirements are going to be met: verify identity, get shared courses, register shared courses for external learning systems and publish/remove web services for administrators. The final class design is determined in this section by means of three types of approaches: *Package View*, *Class View* and *Component View*.

#### 4.5.3.1 Package View

Packages are used to represent groups of classes and serve to partition the logical model of an application. [31] The following figure illustrates the packages and their relations of the system:

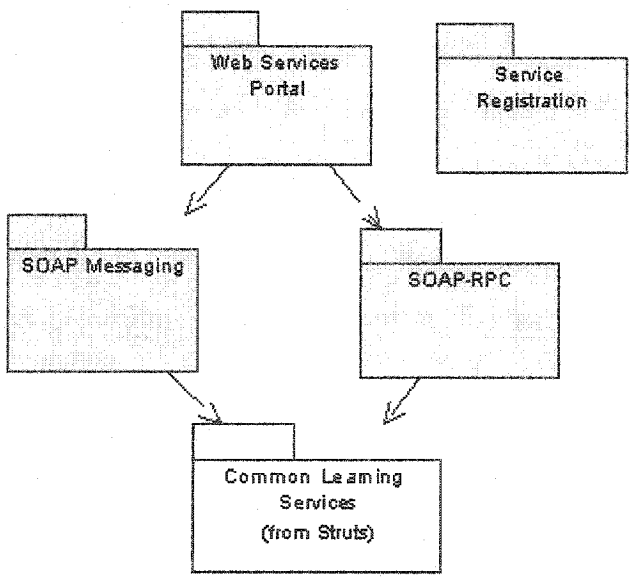



**Figure 4-19** System Package View

There are five main packages:

- Web Services Portal Package: it contains the remote interface classes for external learning systems to access Web services and also contains a UDDI query class.
- SOAP Messaging Package: it contains the Web service request client, the request message handler and the Web services using SOAP messaging mechanism.
- SOAP-RPC Package: it contains the RPC service client, the RPC service interface and the RPC service using SOAP RPC mechanism.
- Common Learning Services Package: it contains the classes of implementing the actual learning services that the service requesters are ultimately interested in accessing.
- Service Registration Package: it is an independent package that contains the classes for publishing and removing Web services.

#### 4.5.3.2 Class View

Based on the package view, the final class diagram is like this:

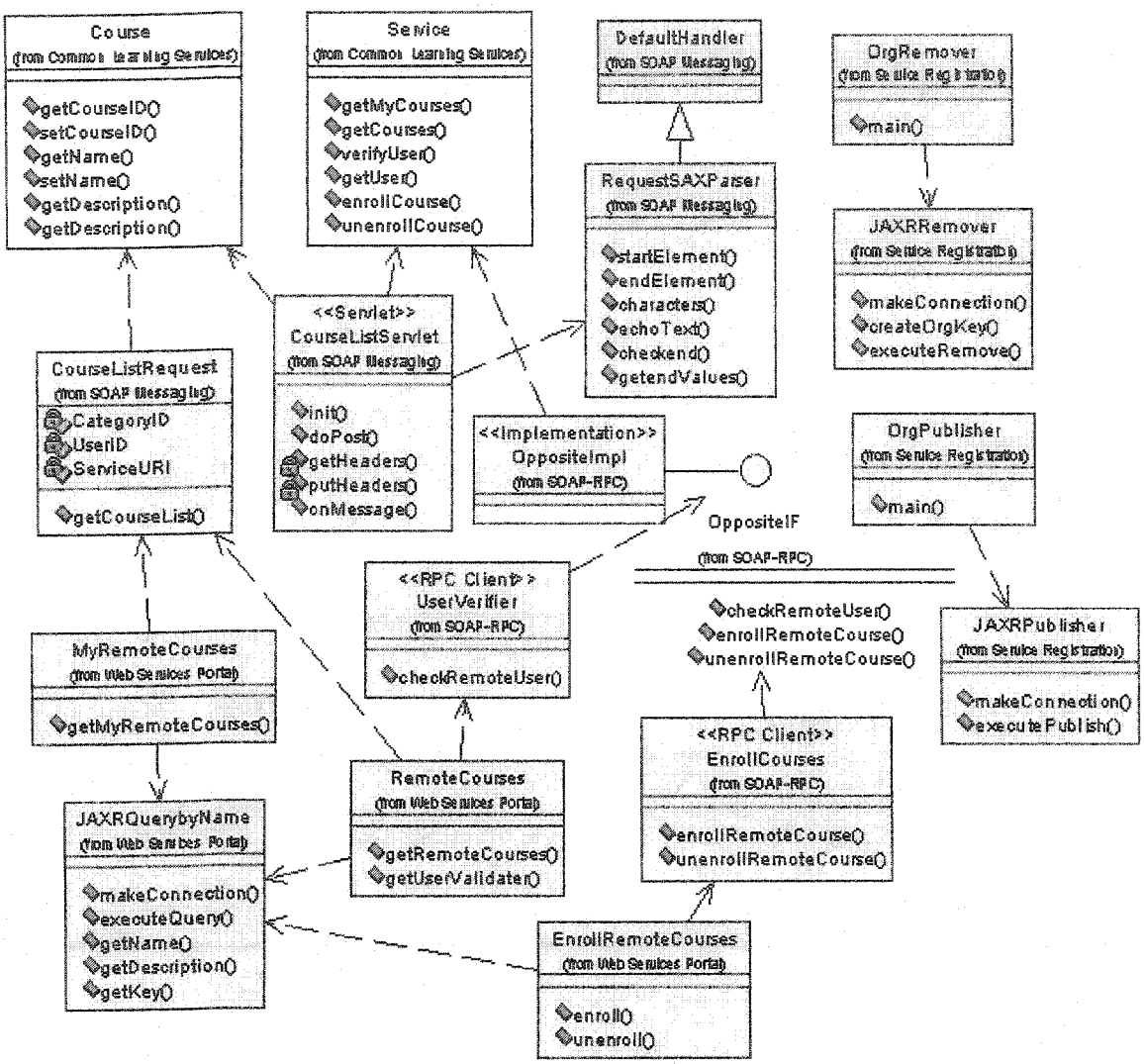



**Figure 4-20** System Class Diagram

Figure 4-20 shows how the main functions are addressed properly:

- JAXRQuerybyName: the class for searching Web services from a UDDI registry using Java API for XML Registries (JAXR).
- RemoteCourses: the portal class for getting the shared course information through SOAP messaging.
- MyRemoteCourses: same as RemoteCourses class but for a specific registered course only.
- EnrollRemoteCourses: the portal class for registering and deregistering courses through SOAP based RPC.
- UserVerifier: the RPC client class for user identity verification.

- EnrollCourses: the RPC client class for registering remote courses.
- OppositeIF: the interface providing RPC based Web services using Java API for XML-based RPC (JAX-RPC).
- OppositeImpl: the implementation of the interface OppositeIF.
- CourseListRequest: the client class for sending course information request in a SOAP format.
- CourseListServlet: a HTTP servlet class, in which there is the SOAP messaging-based Web service for providing shared course information using Java API for XML Messaging (JAXM).
- RequestSAXParser: a parsing class based on the standard SAX (Simple API for XML Parsing) for extracting user request data from the SOAP request message.
- OrgPublisher, OrgRemover: the classes for publishing Web services to UDDI registry and removing from UDDI registry respectively.
- Course: the course class with its main properties.
- Services: the interface of the ultimate common learning services for the Web services tier to invoke.

#### 4.5.3.3 Component View

Components are vertical groups of classes with behavioural proximity at a physical level. [31] Figure 4-21 shows the component view of the system.

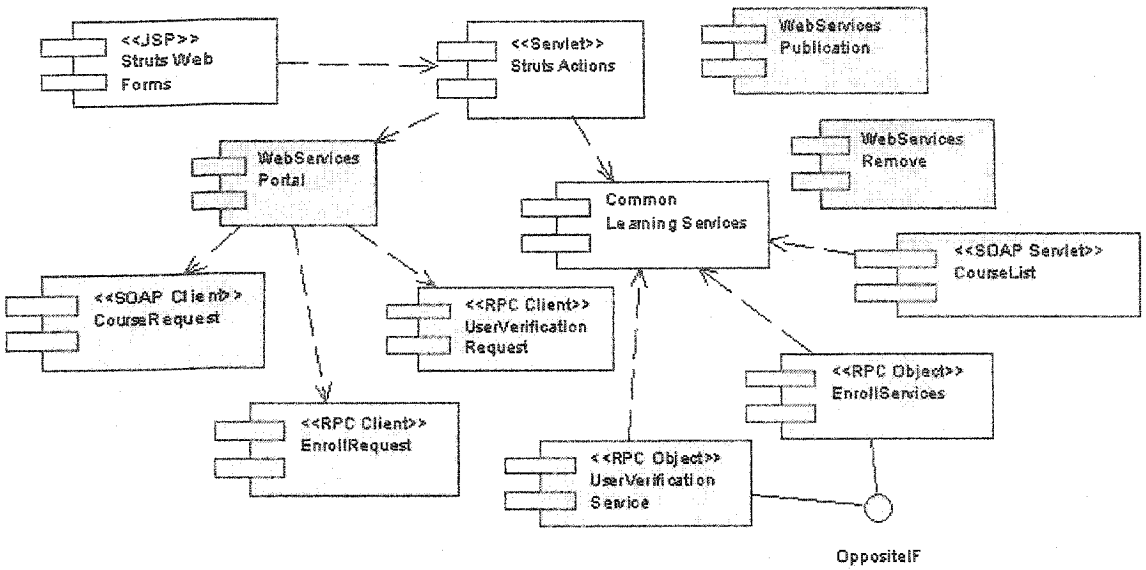



**Figure 4-21** System Component Diagram

Figure 4-21 clearly demonstrates how those Web services components (in deep colour) are related to construct the Web services tier of our system architecture. Once a service request is sent from the Struts-based web forms, the corresponding Struts action is waken up to either call the common learning service locally or dispatch the request to the Web services portal component to invoke the remote service of other LMS like user verification or course registration by instancing one of the Web service client components. On the other hand, the system's learning services facing to external learning systems are ready to be invoked with their RPC interfaces or HTTP servlet portals. The Web services publication (remove) component is responsible to publish (remove) those services binding information such as its URI and its WSDL document into (from) a UDDI registry.

# Chapter 5 System Implementation: UbiLearn

UbiLearn is a wired/wireless accessible distributed learning system based on the Web services framework, which is designed following the previous models and implemented by the Java compatible technologies at the Multimedia Communication Research Laboratory, University of Ottawa. This chapter describes how the Web services in UbiLearn are implemented, the essential technologies for building Web services, the technologies for accessing Web services through wireless devices and a sample of cross-platform Web service invocation is also introduced.

## 5.1 Implementation Architecture

An overall system architecture is described in section 4.5.1. The architecture in this section is more focusing on the implementing flows between different layers.

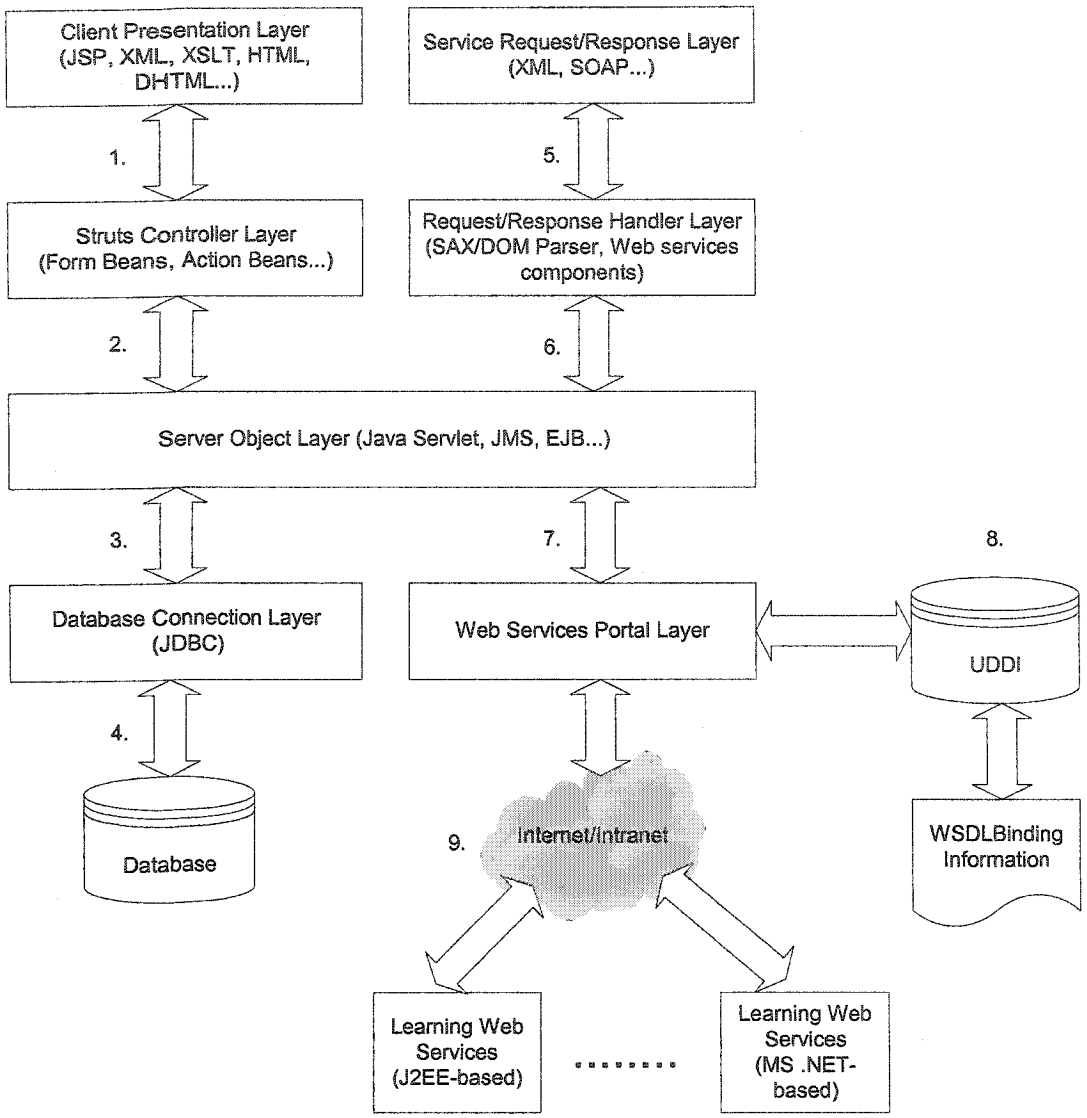



**Figure 5-1** Implementation Architecture of the System

Flows 1, 2, 3 and 4 are the normal ways how a Web-based learning system works.

1. From the Client Presentation Layer (e.g. a Internet browser or a Web enabled wireless device), in which the client contents are presented by the forms of HTML, JSP (Java Server Pages) or XML, learner or instructor can submit their information through HTTP requests to the Struts Controller Layer and get HTTP responses.
2. Two basic models of the Struts design pattern: Form Beans and Action Beans, which systematically represents the user request data and user actions, dispatch

the request data to a specific Server Object such as a Java Servlet, a Java Message Service (JMS) or a Enterprise Java Bean (EJB).

3. The Server Objects use the Database Connection Layer like JDBC (Java Data Base Connection) API to get information from the database.

Flows from 5 to 9 indicate the interactions between layers of a Web services-based learning system.

5. A service requester sends SOAP requests and gets SOAP responses over HTTP.
6. The Request/Response Handler Layer parses the request streams using the Simple API for XML Parsing (SAX) or Document Object Model (DOM) and transmits the data to the corresponding Web services components. Once the data is extracted from the request stream, the same Server Objects will be invoked by the Web service components.
7. If a Server Object needs a remote learning object or service, it can locate and invoke the object or service through the Web Services Portal Layer where the Web service clients reside.
8. By doing a look up in a UDDI registry, the Web Service Portal can search and locate the desired learning service. The location of and WSDL binding information will be sent back as a SOAP message. The binding information of your own sharable learning services is also published here.

After the service portal gets the binding information, it can directly invoke the learning service by passing the essential data indicated by the WSDL file in a SOAP message over the Internet or Intranet. And the learning service could be J2EE-based or MS .NET-based residing on any platform.

## 5.2 Java Enabled Web Services Developing APIs

To implement Web services components, there are some options to choose: Perl, Python or Microsoft .NET. But why do we choose Java? The simple answer is that Java is platform neutral and it provides a very good support to XML: Java APIs for XML (JAX), which contains Java API for XML Processing (JAXP), Java API for XML-based RPC (JAX-RPC), Java API for XML Messaging (JAXM) and Java API for XML Registries (JAXR). JAX coming with the J2EE SDK 1.4 or Java Web Services

Development Package (JWSDP) 1.0 [34] allows you to write web applications entirely in the Java. The most important feature of the Java APIs for XML is that they all support industry standards, thus ensuring interoperability. Various network interoperability standards groups, such as the World Wide Web Consortium (W3C) [35] and the Organization for the Advancement of Structured Information Standards (OASIS) [36], have been defining standard ways of doing things so that businesses that follow these standards can make their data and applications work together. The flexibility is also another important feature of JAX. Developers have flexibility in how they use the APIs. For example, JAXP code can use various tools for processing an XML document, and JAXM code can use various messaging protocols on top of SOAP.

The implementations of Web services of UbiLearn fall into three major categories: *XML Handling API*, *Service Invoking API* and *Service Sharing API*.

### 5.2.1 XML Handling API

Since the XML is the glue among Web services components, a handy tool for XML parsing and composing is necessary. The class named "RequestSAXParser" in UbiLearn project is doing this kind of job. The Java API for XML Processing (JAXP) leverages the parser standards SAX (Simple API for XML Parsing) and DOM (Document Object Model) so that different parsing methods can be chosen for different scenario. SAX is an event driven, serial access mechanism for accessing XML documents, which is fast. All the callback methods are defined specifically, and the parser invokes them as it reads the XML data. It is so efficient for just retrieving or reading data from XML documents. If we want to manipulate a XML document e.g. modify, remove or insert data to it, DOM is an easier way to do it. DOM is based on the whole structure of XML documents. DOM constructs a hierarchy (tree) structure in the memory after it parses an existing XML. Then we can create nodes, remove nodes, change contents and traverse the node hierarchy using the DOM functions. DOM needs large memories.

Let's take the Listing 5-7 as an example to show how the SOAP request is parsed and composed with the SAX parser ("RequestSAXParser" class in Figure 4-20). Here is the main flow of the process.

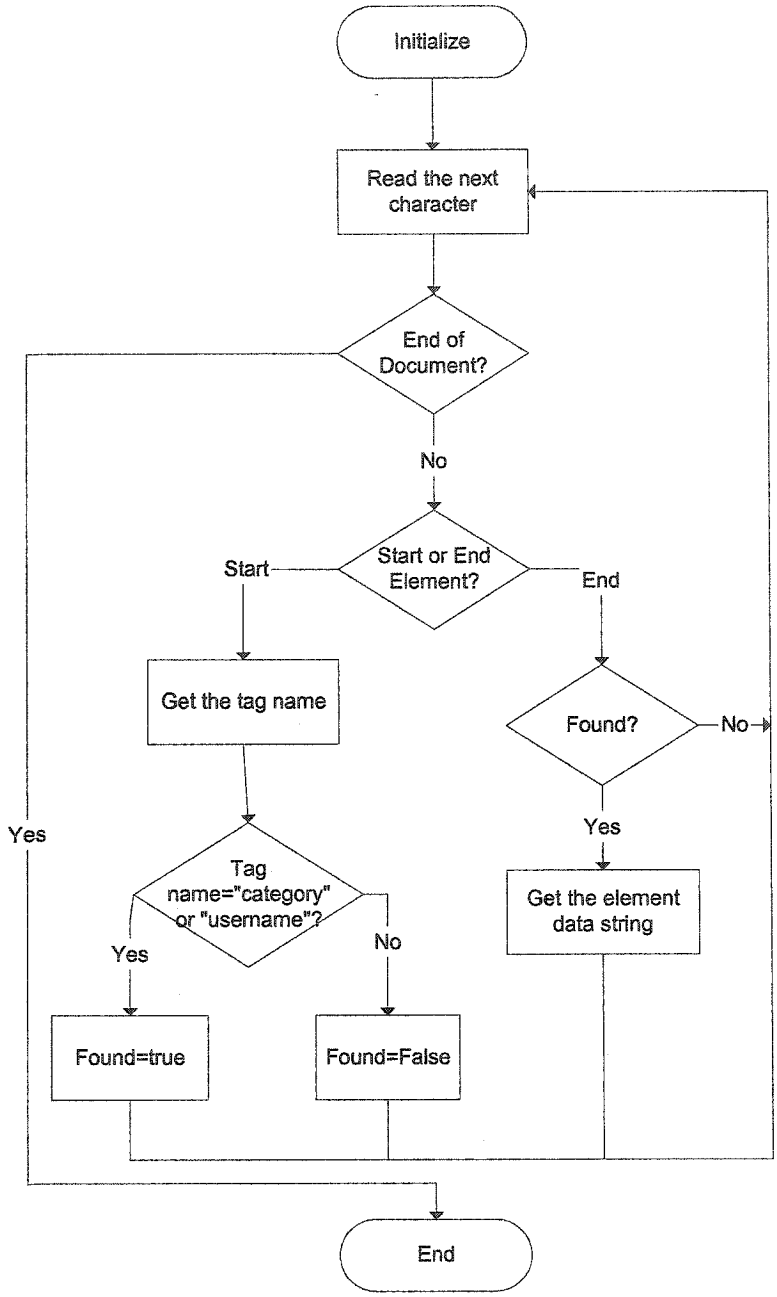



**Figure 5-2** Flow Chart of the SAX Parser "RequestSAXParser"

To retrieve any text content or element attributes of the XML with the SAX API provided by JAXP, the first thing is to create a `SAXParser` object from a `SAXParserFactory` object, then call the method to parse on it, passing the XML source and an instance of a new handler class that is extending `DefaultHandler`. In this handler class, we write our own implementations to `startDocument`, `endDocument`, `startElement`, `endElement` and `characters`. Here are the code segments of "RequestSAXParser":

```
//Initializing:
SAXParserFactory factory = SAXParserFactory.newInstance();
SAXParser saxParser = factory.newSAXParser();
saxParser.parse("request.xml", handler);

//handler Class:
public void startElement(..., String elementName, ...){
if(elementName.equals("category")){
          Found=true;
} else if(elementName.equals("username")){
          Found=false;
      }
}
public void endElement(..., String elementName, ...)
    throws SAXException
    {if (Found){
          echoText();
        }
    }
…
//get the next character
public void characters(char [] buf, int offset, int len) {
String s = new String(buf, offset, len);
     if (textBuffer == null) {
          textBuffer = new StringBuffer(s);
        } else {
          textBuffer.append(s);
        }
}
```

**Listing 5-1** Code Segments of the SAX Parser "RequestSAXParser"

Unlike the SAX parser, building a DOM tree in the memory is easier to access particular pieces of data randomly in an XML document. Once a DOM is tree is built in the memory, you can manipulate it with DOM objects such as `Document`, `Node`,

`NodeList`, etc. With the methods provided by these objects like `createElement`, `appendChild`, `removeChild`, you can insert or remove elements to the XML document, whereas you cannot use SAX parser to do that because it only reads data. JAXP also provides the XSLT (XSL Transformations) API with the `javax.xml.transform` package, which allows plugging in XSLT transformer to perform transformation from XML document to HTML using XSL stylesheet. It is not necessary to describe all the details here.

### 5.2.2 Service Invoking API

Once a Web service is available on the Internet, there are two standard ways to invoke it, which are *SOAP Messaging* and *SOAP-RPC*. JAXP provides the Java API for XML messaging (JAXM) and the Java API for XML-based RPC (JAX-RPC) to implement the two different mechanisms respectively.

#### 5.2.2.1 SOAP Messaging

SOAP messaging is a standard way to send XML documents over the Internet and it is based on the SOAP 1.1 and SOAP with Attachments specifications that are proposed by the World Wide Web Consortium (W3C). There are two kinds of SOAP messaging: request-response messaging (synchronous messaging) and one-way messaging (asynchronous messaging).

In the request-response messaging case, the requester using JAXM API is limited to sending point-to-point messages directly to a Web service. Request-response message is synchronous meaning that a request is sent and its response is received in the same operation. In contrast, the one-way messaging makes the JAXM client totally unaware of how the messages are routing and responding because there is a message provider taking care of assigning message identifier, storing messages and keeping track of whether a message has been delivered before. Figure 5-3 shows the tow kinds of messaging:

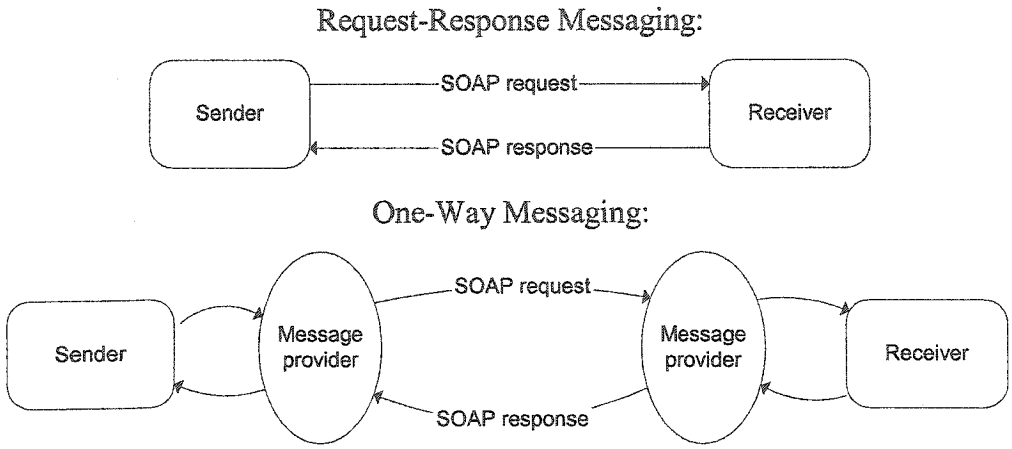



**Figure 5-3** Request-Response Messaging and One-Way Messaging

An interesting feature of the one-way messaging is the unawareness of JAXM client. You just write your JAXM application, and use the `ProviderConnection.send` to send the message. The message arrives at its final destination without knowing about the details involved in accomplishing the delivery. The messaging provider gives the guaranteed delivery. A JAXM client may make a connection with its provider and send one or more messages, then close the connection. The provider will store the messages and send them. It will resend a message that was not successfully delivered until it is successfully delivered or the number of resends is reached. But whether to use a messaging provider or not depends on the specific circumstance. If the JAXM based Web service wants to be either a sender or receiver and can be able to get and save requests that are sent at any time, a messaging provider is more appropriate to use.

A JAXM message is made of two parts, a required SOAP part and an optional attachment part, which consists of a `SOAPEnvelope` object containing a `SOAPHeader` object and a `SOAPBody` object. The JAXM API is defined in two specifications: 1) SAAJ (SOAP with Attachments API for Java) 1.1 – the `java.xml.soap` package, which is for creating SOAP messages, adding message content and extracting message content. 2) JAXM 1.1 – the `javax.xml.messaging`, which is for sending messages with synchronous or asynchronous way. In the UbLearn project, the "get course information" service created by JAXM is actually a Java Servlet that accept HTTP request and send HTTP response in a synchronous messaging way.

### 5.2.2.2 SOAP-RPC

A SOAP-RPC (SOAP - Remote procedure Call) based Web service is a collection of procedures that can be called by a remote client over the Internet. An RPC mechanism enables clients to execute procedures on other systems. In JAX-RPC, a RPC call is also based on SOAP that means although JAX-RPC relies on complex protocols, the messages exchanged between caller and callee are all SOAP formatted. The standard SOAP interface allows a JAX-RPC client to access a Web service that is not running on a Java platform and vice versa. Here is the JAX-RPC runtime structure as specified in Figure 5-4:

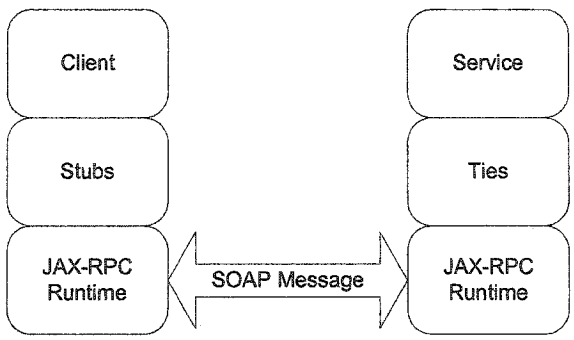



**Figure 5-4** JAX-RPC Runtime Mode

When a JAX-RPC client wants to call a remote service, it invokes a method on a stub, which is a local object represents the remote service. The stub triggers the JAX-RPC runtime systems and converts the remote procedure call into a SOAP message then sends the messages as an HTTP request. When the server receives the SOAP request, the runtime extracts the message from it and translate it to a method call on a tie object. The tie object invokes the method of the service then the runtime on the server converts the method response into a SOAP message and sends it back to the client as an HTTP response. The runtime on the client side extracts the SOAP message from the HTTP response and then translates it into a method response for the client program. The following is a code example of a JAX-RPC service, which is actually a Java interface (see "OppositeIF" in Figure 4-20) for verifying a remote user (the implementation of this interface is not shown below) used in the UbiLearn project.

```
import java.rmi.Remote;
```

```
import java.rmi.RemoteException;

public interface OppositeIF extends Remote {
     public int checkRemoteUser(String userid, String password)
throws RemoteException;
}
```

**Listing 5-2** Code Segments for JAX-RPC Client Interface "OppositeIF"

The actual SOAP request message converted by the stub object for invoking this service is shown below:

```
<?xml version="1.0" encoding="UTF-8"?>
   <env:Envelope
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://opposite.org/types"
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <env:Body>
         <ans1:checkRemoteUser
xmlns:ans1="http://opposite.org/wsdl">
            <String_1 xsi:type="xsd:string">kelvin</String_1>
            <String_2 xsi:type="xsd:string">111</String_2>
         </ans1:checkRemoteUser>
      </env:Body>
   </env:Envelope>
```

**Listing 5-3** The SOAP Request Message for Invoking the JAX-RPC Service "checkRemoteUser"

As can be seen that the method name and parameters for the remote procedure are converted into some corresponding SOAP elements. And the SOAP response message from the RPC service is like this:

```
<?xml version="1.0" encoding="UTF-8"?>
   <env:Envelope
```

```
xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:enc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:ns0="http://opposite.org/types"
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
      <env:Body>
         <ans1:checkRemoteUserResponse
xmlns:ans1="http://opposite.org/wsdl">
            <result xsi:type="xsd:int">3</result>
         </ans1:checkRemoteUserResponse>
      </env:Body>
   </env:Envelope>
```

**Listing 5-4** The SOAP Response Message to the JAX-RPC Client

Comparing to the SOAP messaging communication mechanism, the SOAP-RPC focuses more on being easy to use for the most Web service common tasks that are based on point-to-point communication. A JAX-RPC client simply makes Java remote calls without taking care of all the transmission details like marshalling and unmarshalling. But for some more heavy-duty tasks such as reliable messaging, routing messages to more than one party and one-way messaging, the JAXM is always a better choice for the message-driven interaction.

### 5.2.3 Service Sharing API

Once a Web service is ready, business should register it to a standard registry server such as UDDI, where its services can be shared and retrieved by other potential business partners and also business can discover other Web services. The Java API for XML Registries (JAXR) provides a convenient way to access business registries over the Internet. The main functionalities JAXR provides are registering a business and searching registry.

With the `BusinessLifeCycleManager` object, a JAXR client can create an organization by its name, its description, its classification within the North American Industry Classification (NAICS) and so on. Once the `Organization` object is created,

the service name, description, binding and endpoint (the URL for the service) can be added in. Finally a Universally Unique ID (UUID) is generated for later reference (modification or being removed). With the `BusinessQueryManager` object, a JAXR client can find an organization by its name or classification and retrieve its services information like the access endpoint. JAXR also supports using an SQL query to search a registry, which is done by a `DeclarativeQueryManager` object.

The messaging mechanism between a JAXR client and a registry is done by JAXM running on top of SOAP, which ensures the interoperable communication. This is done completely behind the scenes, so it is entirely transparent for JAXR users.

## 5.3 Wireless Devices Access Web Services

From the above description, we see that a Web service oriented e-learning system like UbiLearn can be implemented in a relative easy and effective way taking advantage of the JAX package on the J2EE developing and deploying platform. But for most wireless devices with small memory size and low processing speed like cell phones, PDAs or other handheld devices to access Web services, a more efficient resource-saving technology is necessary for handling XML/SOAP at the client side's micro platform such as K Virtual Machine (KVM). The Java 2 Platform, Micro Edition (J2ME) [37] is a set of technologies and specifications developed for small devices. J2ME uses subsets of J2SE (Java 2 Platform, Standard Edition) components such as smaller virtual machines and learner APIs. However, the J2ME lacks standard XML APIs, which even are not included in the coming MIDP 2.0 specification either. Therefore, a third party CLDC libraries that can handle those Web services specific XML protocols are needed. The kSOAP [43] package based on the XML parser for KVM: kXML [42], which are all provided by Enhydra.org is introduced in the following to show how they wok with JAX Web services applications smoothly.

### 5.3.1 Java for Wireless – J2ME

J2ME is a set of technologies and specifications developed for small devices. J2ME uses subsets of J2SE (Java 2 Platform, Standard Edition) components, such as smaller virtual

machines and learner APIs. J2ME is divided into *configurations* and *profiles* in order to span such a variety of wireless devices.

There are two configurations that provide JVMs (Java Virtual Machines) and core API libraries. Connected Limited Device Configuration (CLDC) is for the smallest wireless devices with 160 KB or more memory and slow 16/32 bit processors. CLDC has limited math, string, and I/O functionalities and lacks features. As a result, CLDC virtual machines cannot support most J2SE standard libraries.

Connected Device Configuration (CDC) is for more capable wireless devices with at least 2 MB of memory and 32 bit processors. CDC supports a fully featured Java 2 VM and therefore can take advantage of most J2SE libraries.

A profile builds on a configuration but adds more specific APIs to make a complete environment for building applications. While a configuration describes a JVM and a basic set of APIs, it does not by itself specify enough detail to enable you to build complete applications. Profiles usually include APIs for application life cycle, graphical user interface (GUI), and persistent storage. Mobile Information Device Profile (MIDP) and PDA Profile, two profiles based on CLDC, target cell phones and PDA devices, respectively.

Based on CDC, the Foundation Profile (FP) provides more utility, security and network functions such as fundamental APIs gleaned from J2SE, including classes and interfaces from `java.lang`, `java.io`, `java.security`, `java.util` and more. Figure 5-5 illustrates the J2ME platform architecture [50].

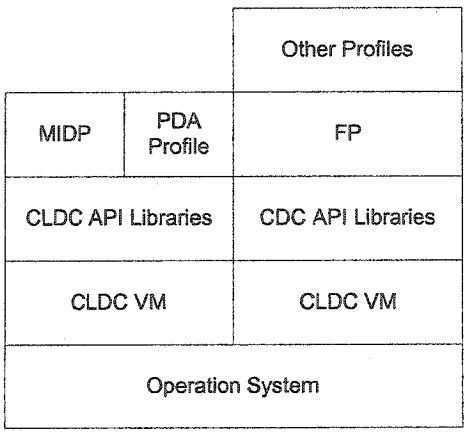



**Figure 5-5** J2ME Platform Architecture

### 5.3.2 Parsing XML with kXML

SOAP is the standard data exchanging protocol for accomplishing the interoperability and heterogeneous accessing in the Web services world. There is no problem for the server side to accept any SOAP request regardless of where it came from because SAX and DOM are powerful XML handling tools. However, these APIs were designed without having the limitation of mobile devices in mind. Lots of features they provide are not acceptable for KVM applications with small size of memory. As we discussed before, SAX parsers need to push all the events such as `startDocument`, `endDocument`, `startElement`, `endElement` to the document processor, then the processor always needs to perform some recovery to determine its own internal processing state before the event handling methods are actually invoked. SAX parsers are "push" parsers.

In contrast to SAX, the kXML API provides a kind of "pull" based parser. It lets the application control the reading data. So the processing state can be implemented much more natural and in recursive functions. The following code fragments demonstrates how the kXML pull parser parses the SOAP request of Listing 5-7 comparing to how the SAX parser does.

```
//initialize
XmlParser parser = new XmlParser ("request.xml");
traverse (parser, "");
//recursive function
public static void traverse (XmlParser parser, String indent)
throws IOException {
      boolean leave = false;
      do {
          ParseEvent event = parser.read ();
          switch (event.getType ()) {
          case Xml.START_TAG:
          if(event.getName().equals("category")){
                        ...
} else if(event.getName().equals("username")){
                        ...
                }
```

```
                traverse (parser, indent + "."); // recursion
                break;
            case Xml.END_TAG:
                ...
                leave = true;
                break;
            case Xml.END_DOCUMENT:
                ...
                leave = true;
                break;
            case Xml.TEXT:
                /*do something to get the text using
event.getText() */
                break;
            case Xml.WHITESPACE:
                ...
                break;
            }
        }
        while (!leave);
    }
```

**Listing 5-5** Code Segments for Parsing XML with kXML Pull Parser

### 5.3.3 Access Web Services with kSOAP

Based on kXML, kSOAP is an SOAP API that leverages the accessing Web services from J2ME platform. The core value of kSOAP lies not only in its ability to retrieve text strings from a SOAP document, which is done by kXML, but also in its ability to map SOAP elements to Java objects. That means the parser can convert the string data extracted from an SOAP message to the related Java data types automatically. kSOAP maps four SOAP types to Java types, which are `Integer`, `Long`, `String` and `Boolean`. Any unknown type is mapped to a `SoapPrimitive` object. The following code segments show how to use `SoapObject` to get the parsed structure of the SOAP response message of Listing 5-8.

```
ByteArrayInputStream bis = new ByteArrayInputStream
(soapRespMesg.getBytes ());
InputStreamReader reader = new InputStreamReader (bis);
XmlParser xp = new XmlParser (reader);
// Use default mapping between Java objects and Soap elements.
SoapEnvelope envelope = new SoapEnvelope (new ClassMap
(Soap.VER12));
envelope.parse (xp);

// Get the parsed structure.
SoapObject course = (SoapObject) envelope.getResult();
// Retrieve the values as appropriate Java objects.
String couseid = (String) course.getProperty ("courseID");
String coursename = (String) course.getProperty ("name");
String duration = (String) course.getProperty ("duration");
String difficultylevel = (String) course.getProperty
("difficultylevel ");
String description = (String) course.getProperty
("description");
String provider = (String) course.getProperty ("provider");
```

**Listing 5-6** Code Segments for Mapping SOAP Elements to Java Types with kSOAP

The automation of data mapping between XML element name and Java types is realized by the parser's `ClassMap` object as we can see from the above code example. Since the kSOAP is a open source project, you can add any custom type mapping and marshaling such as `xsd:dateTime` to `java.util.Date` through the implementation of interface `Marshal`. To compose a SOAP message, we can do it with a reversing processing. First, a `SoapObject` should be built in the memory, then all leaf elements can be appended into the structure.

kSOAP works on serialized SOAP messages for parsing and composing, but how does a J2ME client accesses Web services built by JAX? kSOAP provides a powerful utility class `HttpTransport` that does the help. An `HttpTransport` can be constructed using the JAXM or JAX-RPC endpoint and `SoapAction` URIs (Uniform

Resource Identifiers). Method `HttpTransport.call()` serialized the SOAP message, send it to the message end point and receives the response.

`HttpTransport` could be used to access Web services built by either JAXM or JAX-RPC since both of them run on top of SOAP. The only difference is that for accessing JAX-RPC Web services, the SOAP request messages constructed by the `SoapObject` must follow the structure of Listing 5-3. However, not everything is perfect, although the SOAP response message from JAXM can be converted to a kSOAP's `SoapObject` smoothly, the SOAP request message generated by `SoapObject` cannot be parsed directly by the JAXM's `java.xml.soap` package. That's why an additional XML parser ("RequestSAXParser") is needed at the server side for retrieving the request data. Here is how we did in the UbiLearn project to solve this problem:

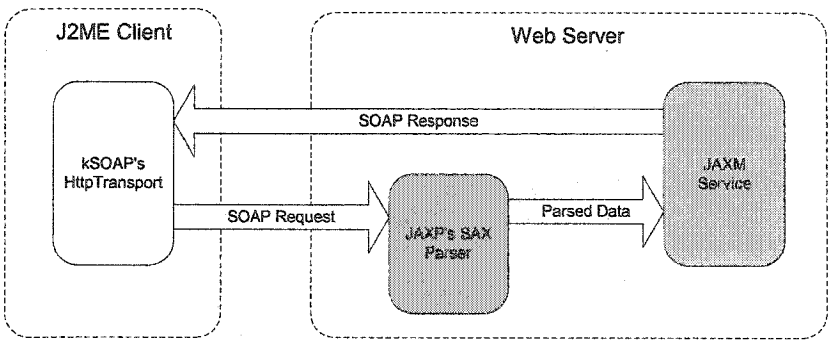



**Figure 5-6** SOAP Request & Response Between J2ME Client and JAXM Web Services

A JAXP's SAX parser being plugged in here for interpreting the SOAP request makes the wireless client and server's interaction more feasible. And the modified structure does not affect any of the original JAXM client's SOAP request message parsing cause the SOAP interface remains unchanged. That also verifies that the flexibility of Web services framework and those components are really loosely coupled.

## 5.4 Sample of Cross-Platform Invocation

After giving detailed description of how to implement the Web services of UbiLearn system according to the analysis and design models, we present a sample of a learning service invocation by different programming languages and within different operation systems to show how Web services are dealing with the application interoperability. The

sample is like this: there is one learning service from the UbiLearn system for providing shared course information, which is implemented by Java technologies and locates on a J2EE based Linux server. A service requester system that is written by Microsoft ASP.NET[9] using C# running on a Microsoft IIS based Windows XP server sends SOAP messages to the learning service and gets the course information as response.

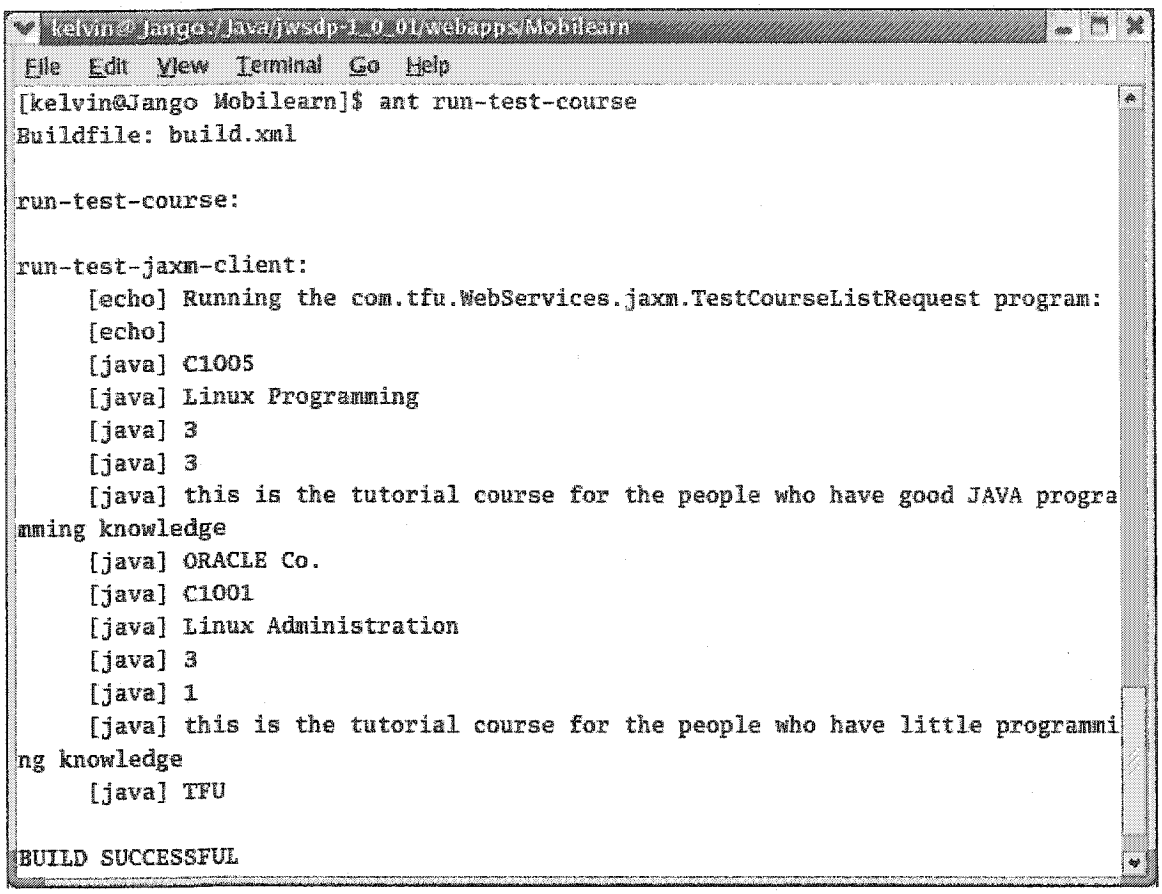



**Figure 5-7** The "Get Course Information" Service Running at a Red Hat Linux 8.0 Terminal

Figure 5-7 shows the "Get Course Information" service output after we run a "TestCourseListRequest" service client program at a Red Hat Linux 8.0 terminal. As can be seen there are two courses provided by this learning system, which are "Linux Programming" and "Linux Administration". The service is developed by the Java APIs for XML (JAX) package and running at an Apache Tomcat 4.0 web container. The

[9] ASP.NET (Active Server Page .NET) is a set of technologies in the Microsoft .NET Framework for building Web applications and XML Web Services.

database server is MySQL 3.23.55 Linux version and is connected to the learning service by MySQL Connector/J 2.0. The following figures are the user interfaces of the service requester system, which just represents the "external learning system".

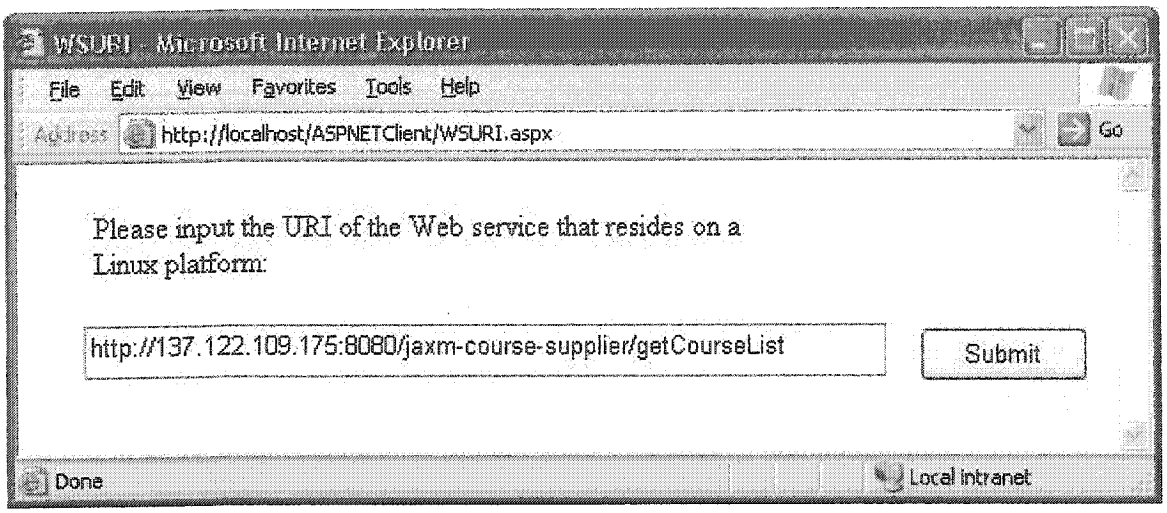



**Figure 5-8** The Service Requester System User Interface 1

In this web page the URI of the Web service that resides on the Linux server should be entered. Usually the URI information is retrieved from a UDDI registry server by the application automatically, but we omit this step for this sample for sake of simplicity. The requester system is an ASP.NET web project based on the Microsoft .NET framework and is written by a new programming language of C#. When we click the Submit button, the C# program just composes a course information request message in a SOAP format and sends it to the service URI over HTTP. Here is the SOAP request:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/">
      <soap-env:Header/>
      <soap-env:Body>
            <RequestCourses:request-courses
xmlns:RequestCourses="http://www.uottawa.ca">
                  <category>1</category>
                  <username>*</username>
            </RequestCourses:request-courses>
      </soap-env:Body>
```

```
</soap-env:Envelope>
```

**Listing 5-7** The SOAP Request From the C# Client on a Windows XP Server

And the result page is shown as follow:

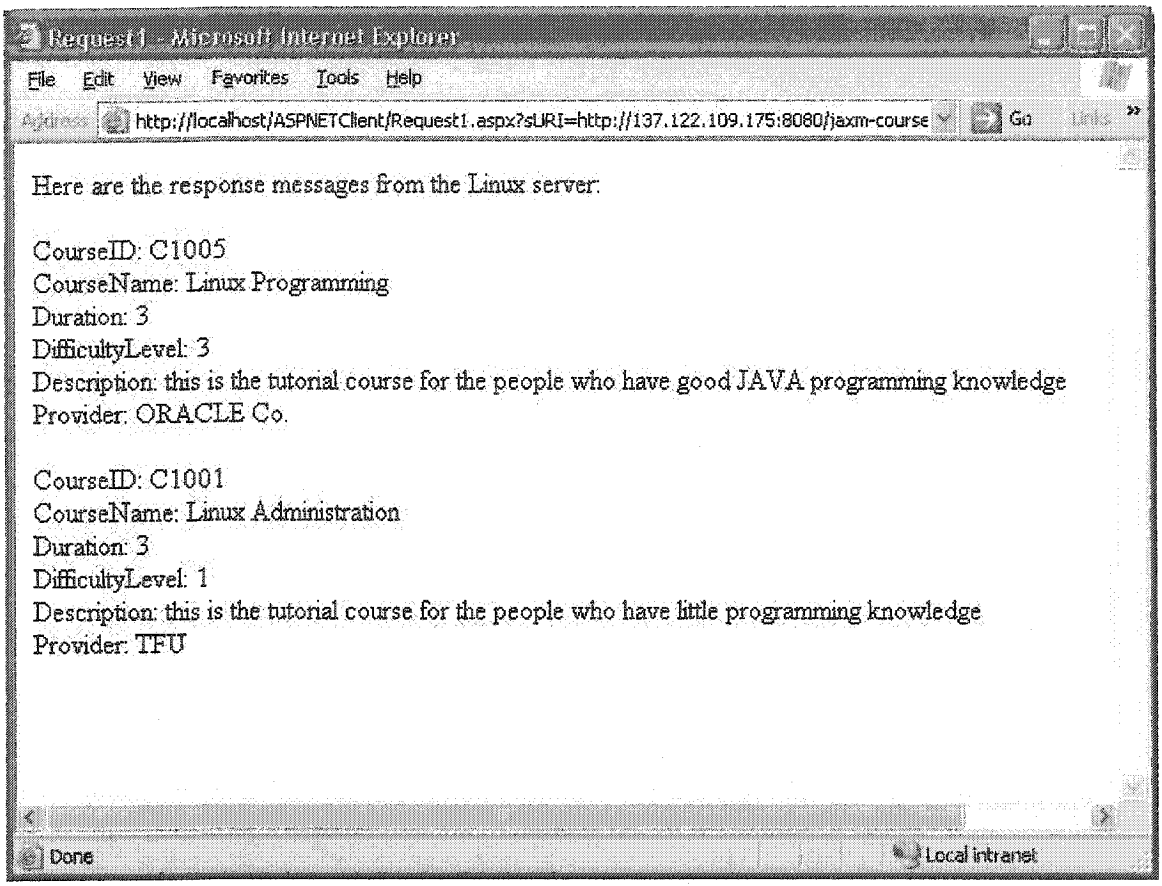



**Figure 5-9** The Service Requester System User Interface 2

This web page shows the same results as we got at the Linux terminal. When the Java servlet gets the SOAP request, it parses the XML stream, extracts the data and invokes the corresponding service regardless of where the message came from. Then it composes a SOAP response message with the course information from the database and sends it back. Here is the response message:

```
<?xml version="1.0" encoding="UTF-8"?>
<soap-env:Envelope xmlns:soap-
env="http://schemas.xmlsoap.org/soap/envelope/" xmlns:SOAP-
ENC="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<soap-env:Header/>
<soap-env:Body>
    <CourseList:course-list
xmlns:CourseList="http://www.uottawa.ca">
        <course>
            <courseID xsi:type="xsd:string">C1005</courseID>
            <name xsi:type="xsd:string">Linux
Programming</name>
            <duration xsi:type="xsd:string">3</duration>
            <difficultyLevel
xsi:type="xsd:string">3</difficultyLevel>
            <description xsi:type="xsd:string">this is the
tutorial course for the people who have good JAVA programming
knowledge</description>
            <provider xsi:type="xsd:string">ORACLE
Co.</provider>
            <courseID xsi:type="xsd:string">C1001</courseID>
            <name xsi:type="xsd:string">Linux
Administration</name>
            <duration xsi:type="xsd:string">3</duration>
            <difficultyLevel
xsi:type="xsd:string">1</difficultyLevel>
            <description xsi:type="xsd:string">this is the
tutorial course for the people who have little programming
knowledge</description>
            <provider xsi:type="xsd:string">TFU</provider>
        </course>
    </CourseList:course-list>
</soap-env:Body>
</soap-env:Envelope>
```

**Listing 5-8** The SOAP Response Message From the Java Web Service on a Linux Server

From this simple example, we can see the Web services has really brought a developing environment that is platform and programming language independent.

## 5.5 Some Useful Tools During Implementation

There are some tools were utilized for monitoring, generating Web services entities during the implementation of UbiLearn. Those tools are so helpful and it is really worth recommending and introducing them here.

### 5.5.1 JAXR Registry Browser

The JAXR Registry Browser comes with the Java Web Services Development Pack (JWSDP). It is both a working example of a JAXR client and a GUI tool that enables you to search UDDI registries and submit data to them. The Registry Browser allows accessing any registry such as the present UDDI registries of IBM and Microsoft and the Registry Server Xindice coming with the JWSDP. The following image shows how the binding information of the Web service "get course information" is found and displayed through the browser. The service is registered at a Xindice Registry Server as one of the Web services provided by the University of Ottawa.

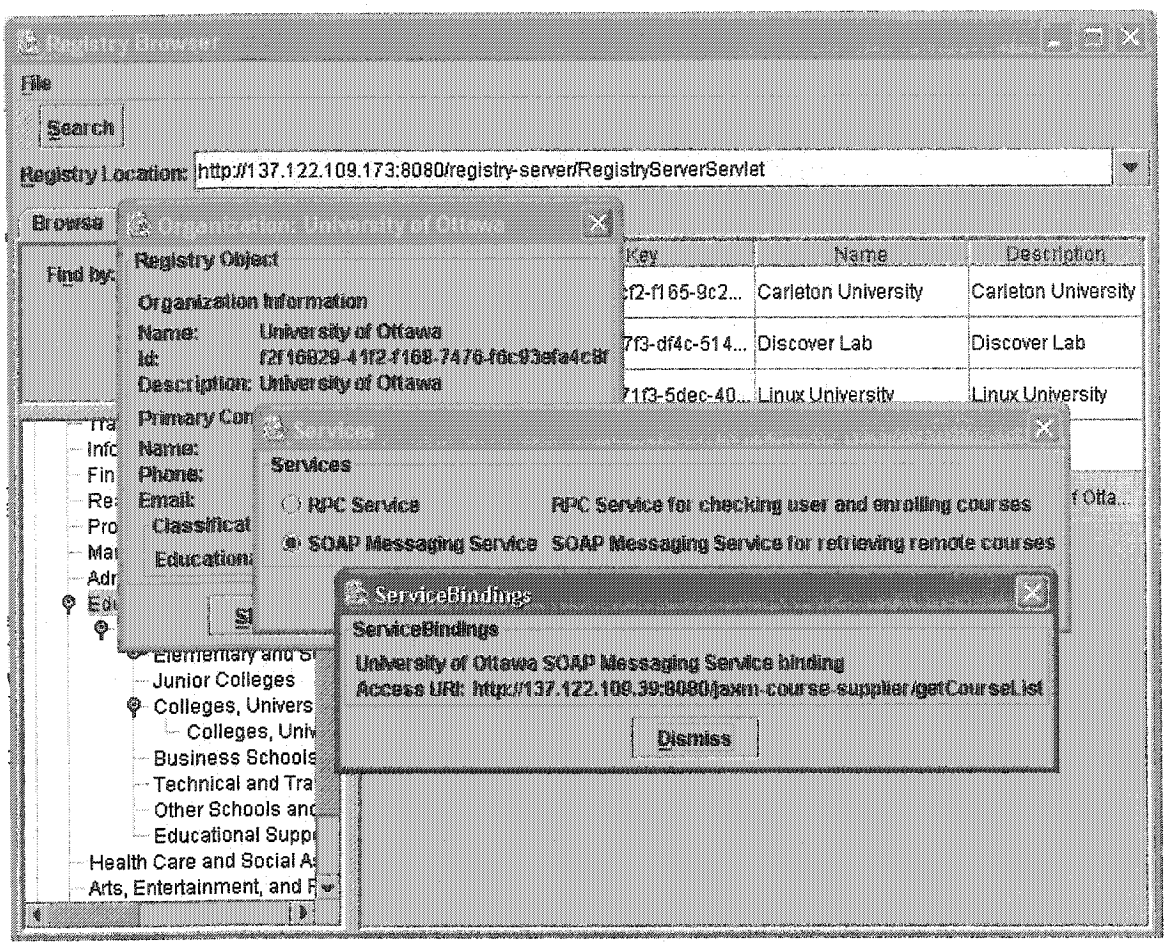



**Figure 5-10** The JAXR Registry Browser

As can be seen, we just need to specify the registry location, enter the keyword of the organization we are looking for and then click search or just search from the classification directory directly, then we can find the binding information of the Web services the organization has published. Also submitting or removing a Web service can be done by the same GUI.

### 5.5.2 TCPMonitor

TCPMonitor is a TCP Monitoring tool, which is provided by the Apache Software Foundation (ASF) [45] and comes with a SOAP Toolkit called AXIS (Apache eXtensible Interaction System). [44] It is so useful for you to monitor the SOAP messages as they are being transmitted between Web service requester and provider. The SOAP request and response of service "get course information" appear in different panels of the TCPMinitor tool, as shown in Figure 5-11.

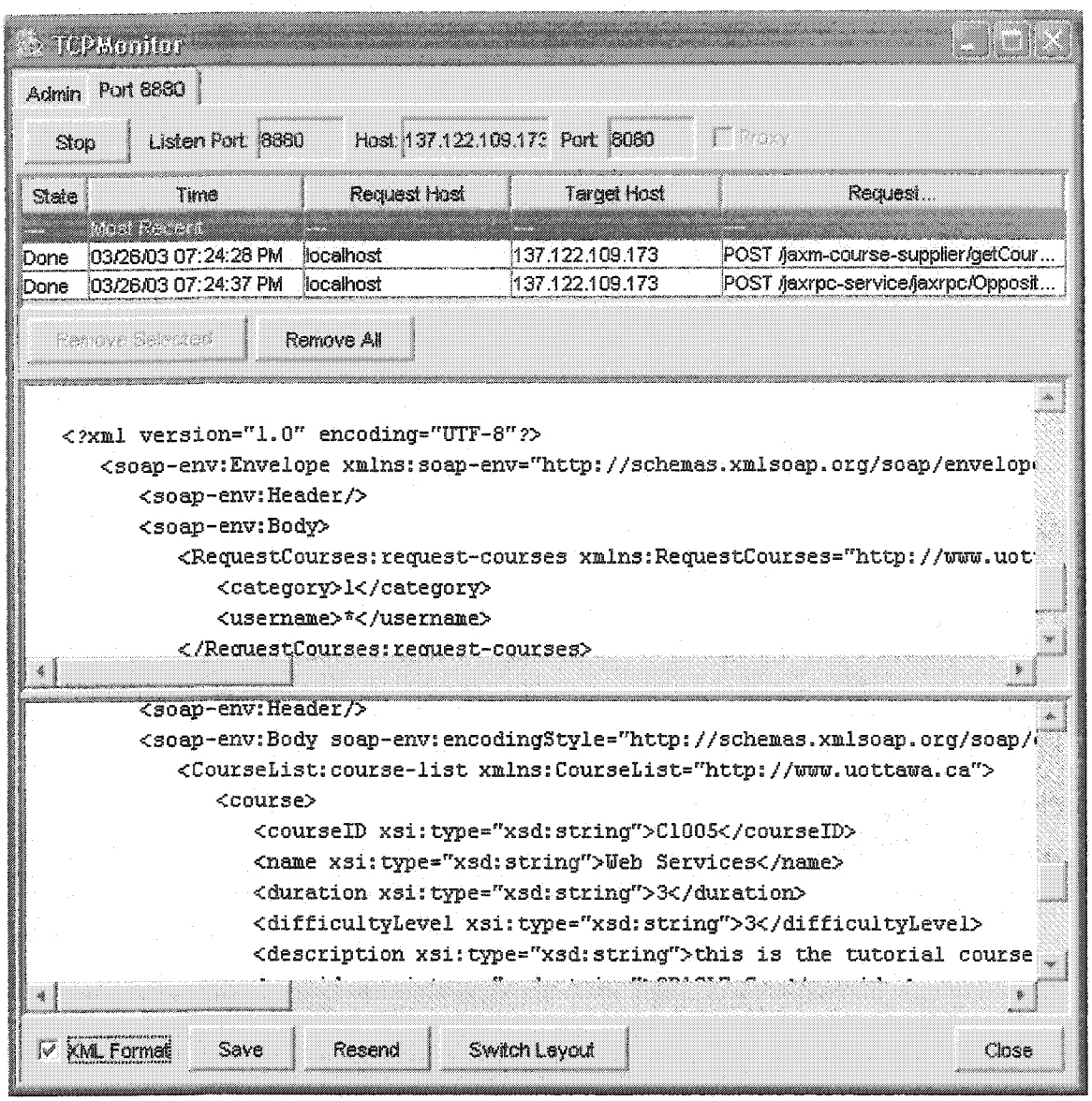



**Figure 5-11** SOAP Message Monitoring Tool: TCPMonitor

With TCPMonitor it is so easy for to know what is going on when we invoke a Web service.

### 5.5.3 Java2WSDL & WSDL2Java Tool

The Web Service Description Language, or WSDL, provides a language-independent technique for describing a Web service to any potential client. A WSDL document allows consumers to access Web services in a standard way. As a service provider we might wish for a way to create a WSDL document simply by introspecting some Java classes. And same as a service requester you might wish for a tool that can generate the client

class to access a service by a given WSDL document. Fortunately, there are two tools that can help us. *Java2WSDL* tool can generate the WSDL file for a given Java class or interface and the other *WSDL2Java* tool can generate the client stub class by taking a WSDL file. Both of them are Java JAR files and can be obtained from the Apache AXIS toolkit.

These tools can be used at a command line and specify some options to the program. Here are some common options for the *Java2WSDL* tool:

- -o, --output: name of the output WSDL file
- -l, --location: the URL of the Web service
- -n, --namespace: the name of the WSDL's target namespace
- -p, --PkgToNS: the mapping of a package to a namespace

The WSDL file of the interface "OppositeIF" generated by *Java2WSDL* is shown in Listing 2-3. Some common options for *WSDL2Java* are listed below:

- -o, --output: the root directory for all emitted files
- -s, --skeleton: in addition to generating the client stub classes, *WSDL2Java* can also create server skeleton classes
- -p, --package: Java package name to place code
- Name of the WSDL file

# Chapter 6 Scalability and Performance Evaluation

Web services are very different to test than desktop software. Scalability and performance are the most important characteristics for evaluating a Web service application. In this chapter, we discuss the methodology for testing the scalability and performance of a specific Web service and analyze the results of the testing.

## 6.1 Testing Overview

Scalability describes a Web service's capability to serve clients under varying levels of load. [11] To measure scalability, we run a Web service test program sending SOAP requests and measure its responding time. Then we run the same test program with 10, 100 and 500 concurrent clients. The test program is running over and over again. Summarizing the measurements enables a software tester to predict the Web service's capability to serve users under load conditions.

Performance is a twin to scalability. Testing one without the other can give you meaningless answers. What good would it be a Web service's ability to serve 1000 users quickly but 200 users receive error response? Scalability testing assumes that valid test clients complete correctly. On the other hand, performance testing measures failures. Performance testing evaluates a Web service's ability to accurately deliver functions. [11] A performance test program looks at the results to see whether the Web service produced an exceptional result.

We already implemented some Web services in UbiLearn project such as the "get course information" service that provides shared course information upon client requests. To test this service, test agents that can send SOAP requests over and over again is needed. TestMaker, which is provided by PushToTest [47], is an open source utility and framework to build and run intelligent test agents. Intelligent test agents implement the typical behavior of archetypical users like keeping on sending requests through standard protocols (HTTP, SOAP, XML-RPC, SMTP) directly to a service. TestMaker offers choices to run test agents:

- Run a single test agent: TestMaker checks a system for correct function

- Run multiple concurrent test agents: TestMaker checks a system for scalability and performance
- Run test agents continuously over time: TestMaker monitors system availability and quality of service.

## 6.2 Test Lab Environment

For testing the "Get course information" service we used three machines hosting the following configurations.

| Machine | Server/device types | Hardware specification | Software specification |
|---|---|---|---|
| Server | Web server & Database server | • Intel Pentium 4 CPU 2.20 GHz<br>• 512M of RAM, 80GB HD<br>• Intel PRO/100 VE Network Card | • Windows XP Server<br>• Sun ONE Application Server 7<br>• MySQL Server 3.23 |
| Machine1 | Client Application Server | • Intel Pentium 4 CPU 2.20 GHz<br>• 512M of RAM, 80GB HD<br>• Intel PRO/100 VE Network Card | • Windows XP<br>• TestMaker 3.3 |
| Machine2 | Client Application Server | • Intel Pentium 4 CPU 2.40 GHz<br>• 512M of RAM, 80GB HD<br>• Alcatel DSL high-speed modem | • Windows XP<br>• TestMaker 3.3 |

**Table 6-1** Testing Configurations for "Get Course Information" Service

The following figure illustrates the network configurations for the testing environment.

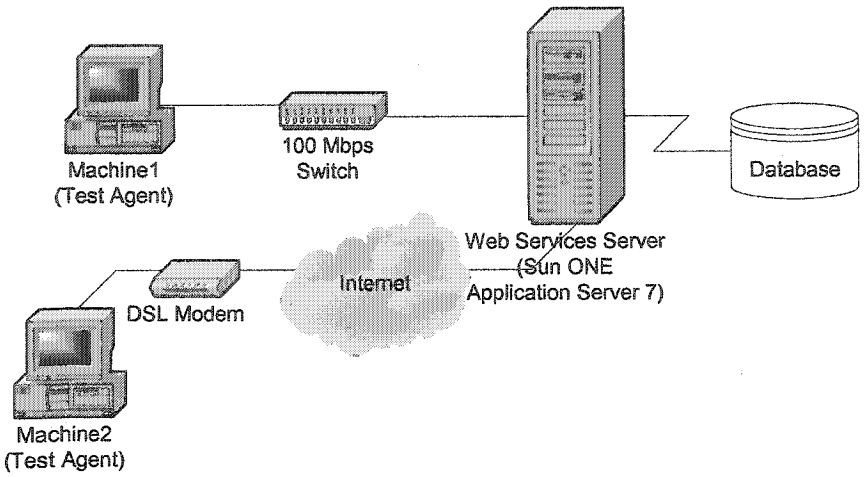



**Figure 6-1** Network Configuration for Testing

The "Get course information" service resides on a Windows XP server that is equipped with the Sun ONE Application Server 7[10]. The service accepts SOAP requests, parses the SOAP messages, then retrieves course data from database and sends back the information in a SOAP format. Machine1 and machine2 are two test clients running multiple concurrent threads of testing agents that compose SOAP requests and send them to the Web services server. Machine1 locates within the same Local Area Network (LAN) as the Web services server does. A 100 Mbps-based 3Com switch connects the server and machine1. The IP of the Web services server is also accessible from outside of the LAN. Machine2 connects to the server by the high-speed Internet access service – DSL (Digital Subscriber Line) provided by Bell Sympatico. The high-speed DSL provides the speeds up to 960 Kbps downstream and up to 120 Kbps upstream. With this kind of network configurations, we can evaluate how different transmission speeds would affect the scalability and performance characteristics of a specific Web services.

The testing results are also relative with the settings of the application server. Here are some configuration settings of the HTTP service form the Sun ONE.

---

[10] Sun ONE Application Server 7 is the new, Java 2 Platform, Enterprise Edition (J2EE) 1.3 compatible application server that combines key Sun sourced technologies to provide a developer friendly yet extremely high performance web services platform.

| | Setting | Description | Value |
|---|---|---|---|
| File Caching | Hash Table Size | The size of hash table for TCP connection lookup. | 2049 |
| | Maximum Age | This setting controls how long cached information will continue to be used once a file has been cached. | 30 seconds |
| | Maximum # of Files | The maximum number of files can be cached. | 1024 |
| Performance Tuning | Maximum Simultaneous Connections | When a new request arrives, the server checks to see if it is already processing the maximum number of requests. If it has reached the limit, it defers processing new requests until the number of active requests drops below the maximum amount. | 1280 |
| | DNS Enabled | It allows you to enable the server to do a reverse lookup of a client's IP in the DNS database before executing a CGI script. | No |
| | DNS Cached Enabled | Determines whether to cache DNS entries. | No |

**Table 6-2** Configurations of the Sun ONE Application Server that hosts Web services

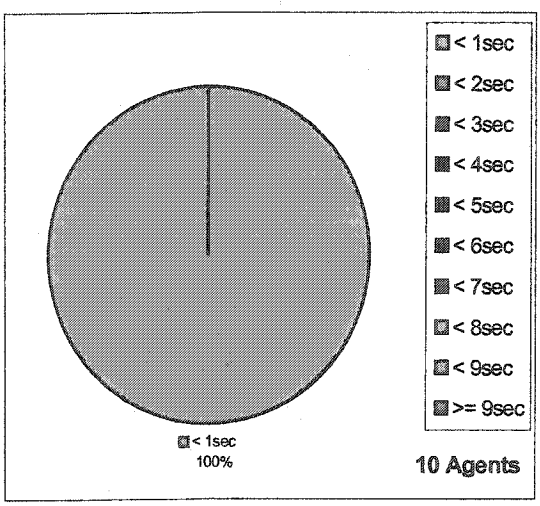
## 6.3 Testing Methodology & Results

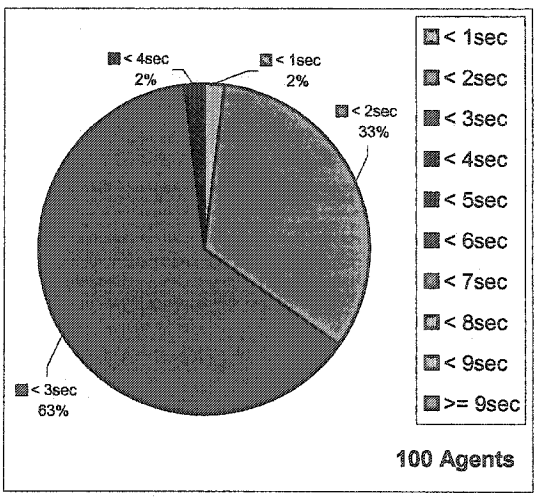
TestMaker 3.3 features an object-oriented scripting language to build intelligent test agents. Using the scripting language, we can run multiple threads of the same test agent to create SOAP connections and send SOAP requests so that the application server thinks there are several clients invoking the "Get course information" Web service concurrently. We test the Web service under different load conditions of 10, 100, 500 and 1000 threads running test agent at the same time respectively. We keep on those threads running over and over again during a specific time period like 1 minute and record the average

response time it took to complete each SOAP request. We hold each response time to an array. The first position in the array holds the response time for requests that took less than one second. The second position holds the response time for requests that took between 1 and 2 seconds, and so on. Then we get the accumulated time in each time slot: [0, 1], [1, 2], [2, 3]... Finally we divide each time value by the total response time of all requests to get the percentages in different time slots. Those percentages just represent the scalability characteristics of the Web service. Table 6-3 and Figure 6-2 show the testing results of the "Get course information" service within the 100 Mbps-based LAN.

| Load Level | < 1sec | < 2sec | < 3sec | < 4sec | < 5sec | < 6sec | < 7sec | < 8sec | < 9sec | >= 9sec |
|---|---|---|---|---|---|---|---|---|---|---|
| 10 Agents | 99.99% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% | 0% |
| 100 Agents | 1.83% | 32.25% | 62% | 1.92% | 0% | 0% | 0% | 0% | 0% | 0% |
| 500 Agents | 0.09% | 0.27% | 0.91% | 1.09% | 4.55% | 8.91% | 10.91% | 7% | 28.57% | 36.27% |
| 1000 Agents | 0% | 0% | 0% | 0.18% | 0.55% | 1.09% | 0.55% | 2.82% | 4.64% | 88.59% |

**Table 6-3** Scalability of the "Get course information" Service within a 100 Mbps-based LAN (in table form)



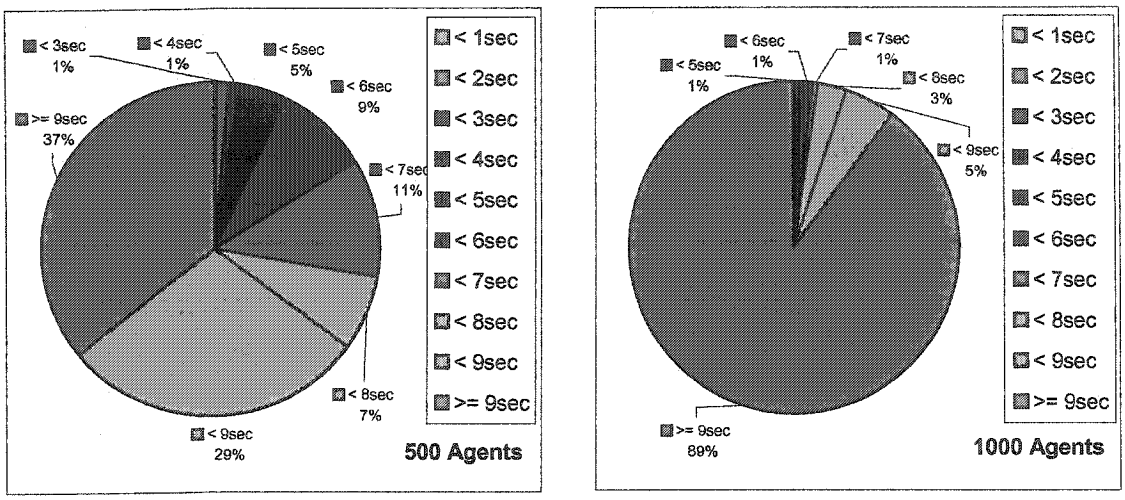



**Figure 6-2** Scalability of the "Get course information" Service within a 100 Mbps-based LAN (in pie form)

The results in Table 6-3 are from the experiments of multiple concurrent threads having run for about 1 minute. In each load level, we record the results every 5 seconds and then calculate the average values. The top line of the table shows the results of running 10 concurrent threads (agents). About 99% of the time the Web service completed the test agents in less than 1 second. For 100 concurrent agents, about 33% of the time the test agents end in less than 2 seconds and about 62% of the time the test agents complete in less than 3 seconds. However, we notice that when more than 500 threads concurrently run the same agent, the Web service does not perform so well. With 1000 agents, about 87% complete the test agent in more than 9 seconds. The test agent number's increasing weakens the scalability of the Web service. The amplitude of the weakness mainly depends on several aspects such as the process capacity of the Web/Application server, the Quality of Service (QoS) policy settings on the server, the efficiency of database access and the bandwidth of the network. Figure 6-2 is another view of the same scalability results.

For performance testing, we use the same testing mechanism but record the number of running exceptions and SOAP fault responses instead of the response time. Table 6-4 and Figure 6-3 show the performance results (run test agents for 1 minute) of the Web service whose scalability was profiled previously.

| Load Level | Number of Success Responses | Number of Error Responses | Total Sessions | Error Response Percentage |
|---|---|---|---|---|
| **10 Agents** | 2311 | 0 | 2311 | 0% |
| **100 Agents** | 3044 | 0 | 3044 | 0% |
| **500 Agents** | 3230 | 243 | 3473 | 7.00% |
| **1000 Agents** | 2932 | 530 | 3462 | 15.31% |

**Table 6-4** Performance of the "Get course information" Service within a 100 Mbps-based LAN (in table form)

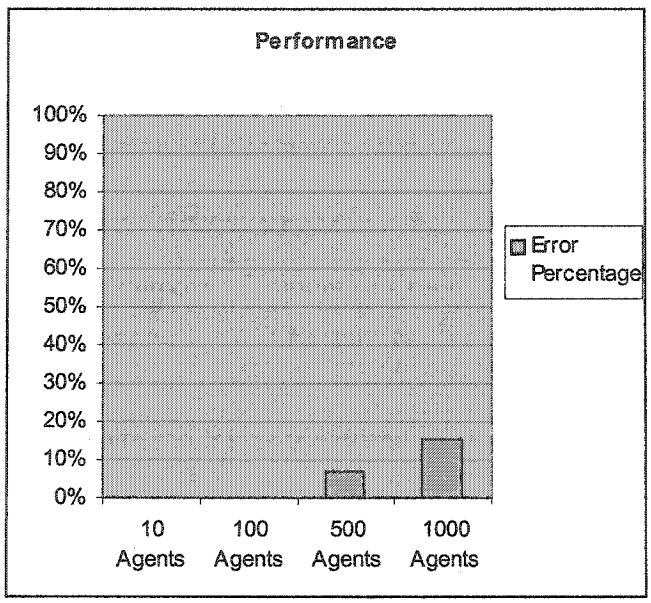



**Figure 6-3** Performance of the "Get course information" Service within a 100 Mbps-based LAN (in column form)

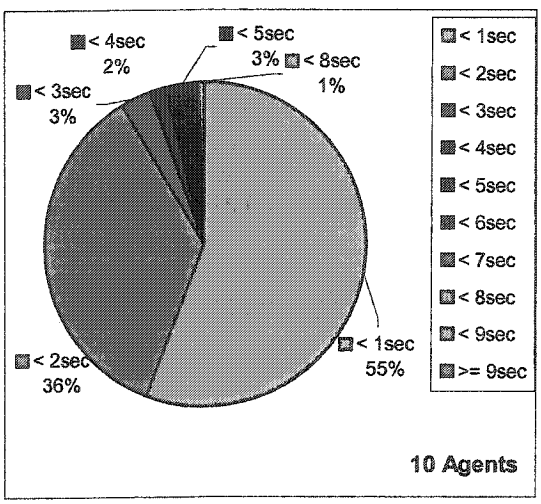
As we can see, the performance results show a different picture of the same Web service. At the 10 and 100 concurrent agent levels, there is no error response during the whole testing period. But there are about 7% and 15% of the requests cannot be completed by the Web service at the 500 and 1000 levels respectively. We found that most of the error responses were "connection refused" exceptions, which means the number of HTTP connection requests to the application server exceeds the maximum value of simultaneous connections that allowed on the server. In other words, the performance of this service is mainly decided by the QoS policy applied on the HTTP server. Certainly the maximum value can be increased to handle more connections but maybe even worse

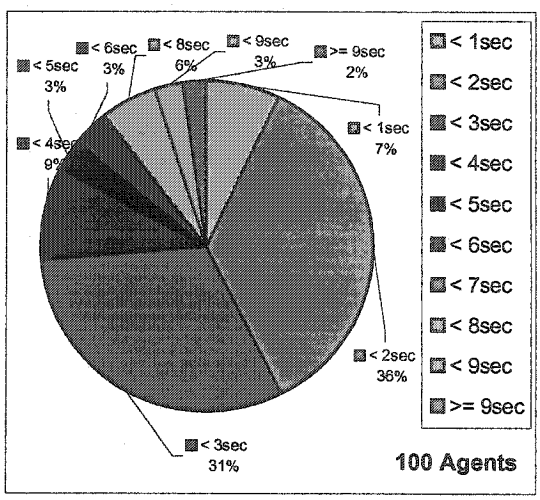
performance or more unexpected errors will occur due to the heavy CPU loading and memory usage on the server.

After studying the scalability and performance of the "Get course information" Web service in the 100 Mbps-based LAN. We wanted to test the performs over the Internet. We used a DSL high-speed modem to connect the Internet service provider and then access to our Web services server. Due to the unsteadiness of the Internet transmission bandwidth (sometimes the speed varies from 50 Kbps at peak time to 500 Kbps at leisure time), we did the scalability test for 8 times under each load condition at different time spots. Then we calculated the averages of those results, which are listed below.

| Load Level | < 1sec | < 2sec | < 3sec | < 4sec | < 5sec | < 6sec | < 7sec | < 8sec | < 9sec | >= 9sec |
|---|---|---|---|---|---|---|---|---|---|---|
| 10 Agents | 54.54% | 35.04% | 2.77% | 1.68% | 3.35% | 0% | 0% | 0.51% | 0% | 0% |
| 100 Agents | 6.15% | 29.69% | 26.18% | 7.86% | 2.57% | 2.95% | 0% | 4.86% | 2.19% | 1.95% |
| 500 Agents | 3.18% | 17.43% | 16.33% | 11.59% | 7.68% | 6.81% | 5.26% | 2.88% | 4.30% | 20.98% |
| 1000 Agents | 0.74% | 6.32% | 8.77% | 6.16% | 8.35% | 7.88% | 6.28% | 7.30% | 6.08% | 35.19% |

**Table 6-5** Scalability of the "Get course information" Service over the Internet (in table form)



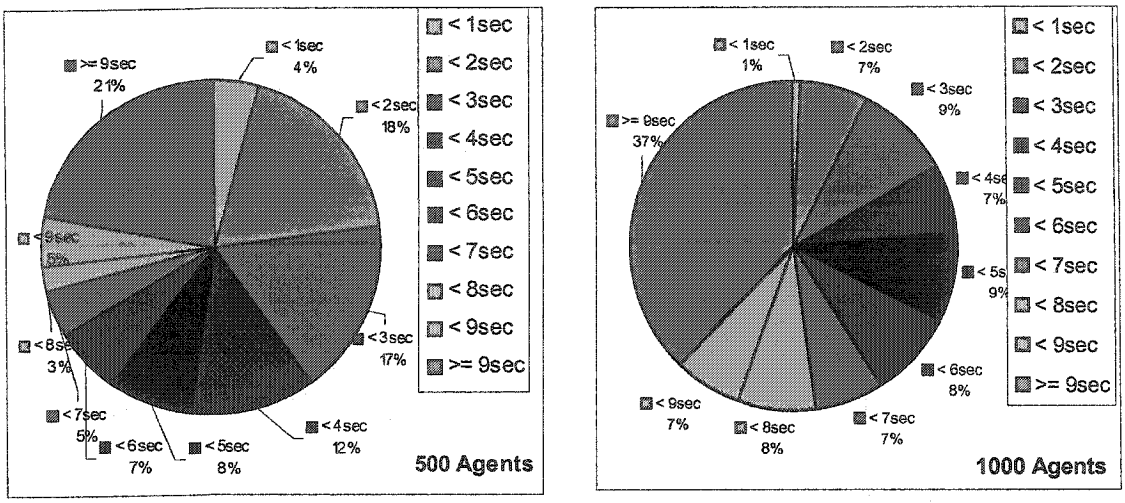



**Figure 6-4** Scalability of the "Get course information" Service over the Internet (in pie form)

Comparing to the test results of the service within the LAN, the scalability over the Internet drops down apparently at the 10 and 100 agents' levels. Only 55% of the time the test agents completes in less than 1 second under the 10 concurrent agents loading condition. And at the 100 agents level the percentages of the time more than 4 seconds are much higher than those in the LAN environment. Just like we expected that the lower bandwidth and the higher latency of DSL Internet access slows down the average response time of the Web service. However, the results of scalability at the 500 and 1000 agents' levels look better than those from the LAN environment. For example, only about 37% of the time the test agents end more than 9 seconds at the level of 1000 agents over the Internet, while about 89% of the time the test agents complete more than 9 seconds at the same level but within the LAN. It looks strange that with the heavy loading conditions even better scalabilities can be achieved over the low-bandwidth Internet than within a 100 Mbps-based LAN. We will find the answer soon. Let's look at the performance results of the testing over the Internet (same as the scalability testing, we did the test 8 times for each load level and got the average values).

| Load Level | Number of Success Responses | Number of Error Responses | Total Sessions | Error Response Percentage |
|---|---|---|---|---|
| **10 Agents** | 520.63 | 6.13 | 526.76 | 1.16% |
| **100 Agents** | 341.38 | 518 | 859.38 | 60.28% |
| **500 Agents** | 184.13 | 1337.63 | 1521.76 | 87.90% |
| **1000 Agents** | 213 | 2469.13 | 2682.13 | 92.06% |

**Table 6-6** Performance of the "Get course information" Service over the Internet (in table form)

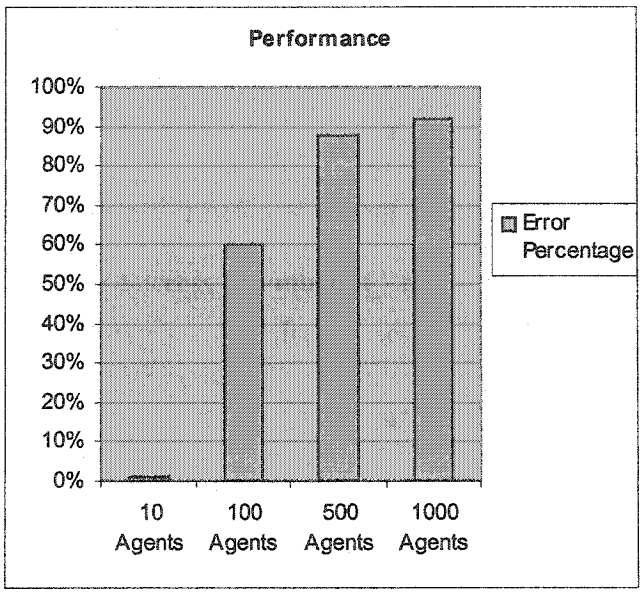



**Figure 6-5** Performance of the "Get course information" Service over the Internet (in column form)

From the test results shown above, it is not difficult to explain that phenomenon. At the relative low loading condition like 10 or 100 agents, the network resources (e.g. network bandwidth, performance policies of the Internet service provider) are able to handle the bunches of simultaneous requests so that the client application still can get most of the accurate responses in a reasonable lower responding time. However, when the loading condition reaches a higher level like 500 or 1000 agents, the performance of the Web service drops down significantly. About 90% of the requests are blocked (connection refused) as 1000 testing agents are keeping on running. Most of the connection requests

are refused even before they reach the Web service server by the Internet service provider according to their performance policy settings to keep the entire network a good load balance, which is the reason why the client application can get responses in a shorter time over the Internet than the LAN at the heavy load levels. While in the LAN environment, in order to process so many concurrent SOAP requests, the application server has to utilize all the resources including CPU, memory and time (slow down the response time) to achieve the relative high performance (low error response percentage).

Comparing those results to those of normal Web applications with traditional distributed methods, we find the learning services based on the Web services framework is acceptable with respect to the scalability and performance characteristics. Normally it is acceptable for a Web service to response a request in no more than 5 seconds at the load level of 10 or 100. The following table summarizes the testing results in two different testing environments.

| | **Load Level** | **Minimum response time** | **Maximum response time** | **Average response time** | **Error rate** | **Requests per second** |
|---|---|---|---|---|---|---|
| LAN | 10 Agents | 0 – 1 second | 1 second | 0 – 1 second | 0 % | 38.5 |
| | 100 Agents | 0 – 1 second | 4 second | 1 – 2 second | 0 % | 50.7 |
| Internet | 10 Agents | 0 – 1 second | 8 second | 0 – 1 second | 1.16 % | 8.8 |
| | 100 Agents | 0 – 1 second | >= 9 second | 1 – 2 second | 60.28 % | 14.3 |

**Table 6-7** Testing Results Summary

From the above results, we can see the average response time is no more than 2 seconds in both two testing environments (only 10 and 100 load levels are listed here because they are the most common load levels for real life web sites and the other two load levels: 500 and 1000 need further testing research and tuning for a better performance). The error rates are so low except the one at 100 levels over the Internet. However, this one

does not make the Web service unacceptable because it depends on the Internet service provider's server settings.

Scalability and performance are measures needed to determine how well a Web service will serve clients. Taken individually, the results may not show the true nature of the Web service. Or even worse, they may show misleading results. Taken together, scalability and performance testing show the true nature of a Web service. [11]

# Chapter 7 Conclusion and Future Research

## 7.1 Conclusions

In this thesis we have presented a Web services oriented framework that facilitates distributed e-learning systems by providing a comprehensive platform in which all the shareable learning objects and contents are published, described, located and invoked in a standardized way. Based on this framework a distributed e-learning system prototype called UbiLearn was analyzed and designed in a new and original way. The prototype was implemented by some powerful Java technologies provided by the J2EE infrastructure like JAXP, JAXM, JAX-RPC and JAXR in a flexible and compatible developing and deploying environment. Also a third party J2ME/CLDC libraries kXML and kSOAP were introduced to show how to handle XML processing and access JAX-based Web services through wireless portal with small memory footprint. Moreover, the scalability and performance tests measures how well the Web service performs under different loading levels. Through this system approach, we realized that most learning services were able to be published and located universally from a UDDI registry and accessed by any application on any platform. And the scalability and performance of the new architecture are still acceptable under most of load levels.

The next generation of distributed e-learning systems requires a sharable place that enables them to launch learning services from different vendors to a common accessible repository and to exchange data between learning applications regardless on what platforms. The Web services technologies fulfill these requirements perfectly. The Web services framework enables systematic application-to-application interactions over the Web, also, a smooth integration of existing infrastructure into the model. The definition of interoperable interfaces and standard communication protocols between application and application are the keys of Web services infrastructure. That enhances the cross-platform development in a lightweight data exchange level. Using the tools described in the thesis, the working prototype has been developed to prove that this kind of Web services oriented e-learning systems can be implemented and different learning services can be shared in a feasible mechanism. Although only three learning services

(namely "get course information", "check remote user" and "enrol remote course") are developed in the system prototype, more learning Web services can be designed and implemented by adapting the idea of those three services. And the testing methodology used in this thesis can be applied to any Web service even with more complicated network configurations. Also the insights resulting from this thesis can be extended to other types of distributed systems like e-business and e-commerce.

Lots of distributed applications have been powered by the Web services technology, but the Web services standards are still undergoing and continuing development pushed by several organizations like W3C, OASIS, IBM, Microsoft, Sun, etc.

## 7.2 Future Researches

The proposed framework has been formulated in the thesis for future research or implementation to be based on. The following research items need further discussion:

- Development of specifications for launching shareable learning object, aggregating and delivering learning resource by using the Web services infrastructure.
- Web services security and access control (authorization & authentication) for learning services' publication and invocation.
- Quality of Services (QoS) is another issue should be focused on related with Web security.
- Scalability and performance improvement for Web services oriented learning applications using SOAP or XML marshalling in the heavy loading conditions.
- Different preferences of accessing learning Web services with different end devices like wired and wireless devices.
- Synchronous and asynchronous collaborative tools for learners and instructors communicating over networks.

# References

[1]. Rama Akkiraju, Henry Chang, Tian Chao, David Flaxer, Jun-Jang Jeng, Frederick Wu, Liang-Jie Zhang. "A Framework for Facilitating Dynamic e-Business Via Web Services": http://www.research.ibm.com/people/b/bth/OOWS2001/akkiraju.pdf, IBM T.J. Watson Research Center.

[2]. Web Services Architecture Working Group of World Wide Web Consortium (W3C): http://www.w3.org/2002/ws/arch/

[3]. Java Remote Method Invocation (RMI): http://java.sun.com/products/jdk/rmi/

[4]. Distributed Component Object Model (DCOM): http://www.microsoft.com/com/tech/dcom.asp

[5]. Distributed Computing Environment (DCE) by the Software Engineering Institute at Carnegie Mellon University: http://www.sei.cmu.edu/str/descriptions/dce.html

[6]. Java Messaging Service (JMS): http://java.sun.com/products/jms/

[7]. Message-Oriented Middleware (MOM) by the Software Engineering Institute at Carnegie Mellon University: http://www.sei.cmu.edu/str/descriptions/momt.html

[8]. Object Management Group (OMG): http://www.omg.org

[9]. Common Object Request Broker Architecture (CORBA) by OMG: http://www.omg.org/gettingstarted/corbafaq.htm

[10]. Common Object Request Broker Architecture (CORBA) by the Software Engineering Institute at Carnegie Mellon University: http://www.sei.cmu.edu/str/descriptions/corba.html

[11]. Robert Brunner, Frank Cohen, Francisco Curbera, Darren Govoni, Steven Haines, Matthias Kloppmann, Benoit Marchal, K. Scott Morrison, Arthur Ryman, Joseph Weber, Mark Wutka. "Java Web Services Unleashed", ISBN: 0-672-32363-X, Sams Publishing, 2002.

[12]. SOAP Version 1.2 Part 1: Messaging Framework by W3C Candidate Recommendation 19 December 2002: http://www.w3.org/TR/soap12-part1/

[13]. SOAP Version 1.2 Part 0: Primer by W3C Candidate Recommendation 19 December 2002: http://www.w3.org/TR/soap12-part0/

[14]. Web Services Description Language (WSDL) Version 1.2 by W3C March 2003: http://www.w3.org/TR/wsdl12/

[15]. Microsoft UDDI Business Registry Node: http://uddi.microsoft.com/default.aspx

[16]. IBM Web Services and UDDI: http://www-3.ibm.com/services/uddi/

[17]. Suppes, P. (1964) "Modern learning theory and the elementary-school curriculum" America Educational Research Journal, 1, 79-93.

[18]. Sharable Content Object Reference Model (SCORM) 1.2 Overview by Advanced Distributed Learning (ADL): http://www.adlnet.org/ADLDOCS/Documents/SCORM_1.2_Overview.pdf

[19]. Sharable Content Object Reference Model (SCORM) 1.2 Content Aggregation Model by Advanced Distributed Learning (ADL): http://www.adlnet.org/ADLDOCS/Documents/SCORM_1.2_CAM.pdf

[20]. Sharable Content Object Reference Model (SCORM) 1.2 Run-Time Environment by Advanced Distributed Learning (ADL): http://www.adlnet.org/ADLDOCS/Documents/SCORM_1.2_RunTimeEnv.pdf

[21]. Lukasz Beca "Applications of XML and Customizable Components in Building Virtual Places on the Web": http://www.collabworx.com/Support/resources/lbpapers/SPDL.html, CollabWorx

[22]. CollabWorx: Integrated Web Collaboration Solutions: http://www.collabworx.com/

[23]. Alliance of Remote Instructional Authoring & Distribution Networks for Europe (ARIADNE): http://www.ariadne-eu.org/

[24]. Institute of Electrical and Electronics Engineers (IEEE) Learning Technology Standards Committee (LTSC): http://ltsc.ieee.org/

[25]. IEEE LTSC Learning Object Metadata (LOM): http://ltsc.ieee.org/wg12/

[26]. Advanced Distributed Learning (ADL): http://www.adlnet.org

[27]. IMS Global Learning Consortium, Inc.: http://www.imsglobal.org/

[28]. Aviation Industry CBT (Computer-Based Training) Committee (AICC): http://www.aicc.org/

[29]. Alan Dennis, Barbara Haley Wixom, David Tegarden. "Systems Analysis and Design", ISBN: 0-471-41387-9, John Wiley & Sons, Inc. 2002.

[30]. Unified Modeling Language (UML) resource page: http://www.omg.org/uml/

[31]. Leszek A. Maciaszek "Requirements Analysis and System Design – Developing Information Systems with UML", ISBN: 0-201-70944-9, Addison – Wesley, 2001.

[32]. Web-Tier Application Framework Design, Designing Enterprise Applications with the J2EE Platform, Second Edition: http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2e/web-tier/web-tier5.html#1080752

[33]. Apache Jakarta Project: http://jakarta.apache.org/

[34]. Java Web Services Development Package (JWSDP): http://java.sun.com/webservices/webservicespack.html

[35]. World Wide Web Consortium (W3C): http://www.w3.org

[36]. Organization for the Advancement of Structured Information Standards (OASIS): http://www.oasis-open.org/home/index.php

[37]. Java 2 Platform, Micro Edition (J2ME): http://java.sun.com/j2me/

[38]. Java 2 Platform, Enterprise Edition (J2EE): http://java.sun.com/j2ee/

[39]. Microsoft .NET Development: http://msdn.microsoft.com/library/default.asp?url=/nhp/Default.asp?contentid=28000519

[40]. Microsoft Web Services Developer Center: http://msdn.microsoft.com/webservices/default.aspx

[41]. Java Technology and Web Services: http://java.sun.com/webservices/

[42]. kXML project provided by Enhydra.org: http://kxml.enhydra.org/index.html

[43]. kSOAP project provided by Enhydra.org: http://ksoap.enhydra.org/index.html

[44]. Apache AXIS: an implementation of the SOAP: http://ws.apache.org/axis/

[45]. Apache Software Foundation: http://www.apache.org/

[46]. Michael Juntao Yuan "Access Web Services from Wireless Devices", Javaworld, http://www.javaworld.com/javaworld/jw-08-2002/jw-0823-wireless.html August 23, 2002.

[47]. PushToTest: http://www.pushtotest.com/

[48]. Francisco Curbera, William A. Nagy and Sanjiva Weerawarana "Web Services: Why and How", IBM T.J. Watson Research Center, August 9, 2001.

[49]. Subrahmanyam Allamaraju "Professional Java Server Programming J2EE Edition", ISBN: 1861004656, Wrox Press, Inc. 2000.

[50]. Michael Juntao Yuan "Java readies itself for wireless Web services", Javaworld, http://www.javaworld.com/javaworld/jw-06-2002/jw-0621-wireless.html June 21, 2002.

# Appendix A: User Interfaces of UbiLearn

## A.1 Main Interfaces for Wired Devices

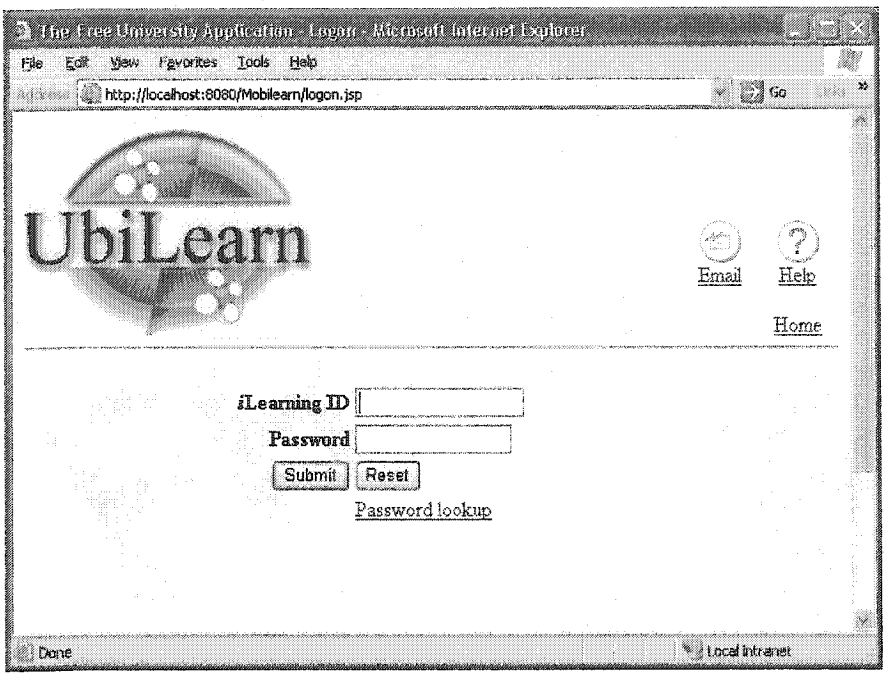



**Figure A-1** User Login

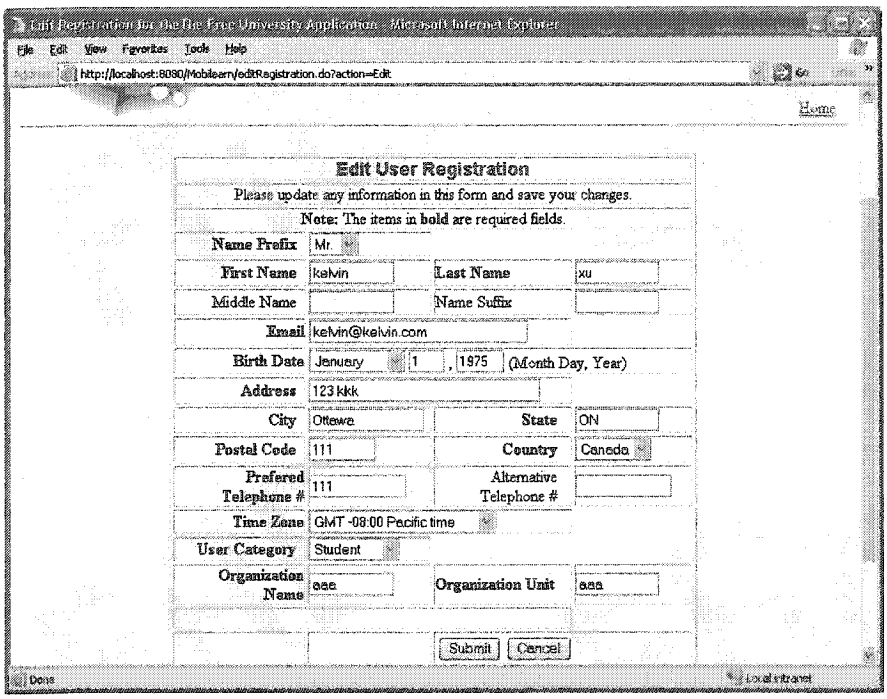



**Figure A-2** User Registration

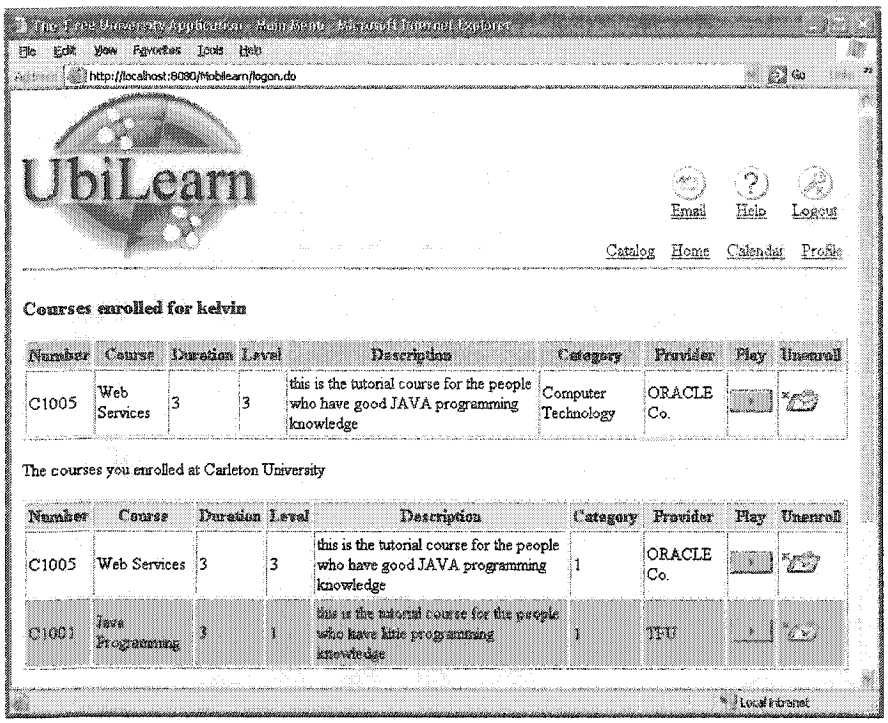



**Figure A-3** Enrolled Courses for Current User

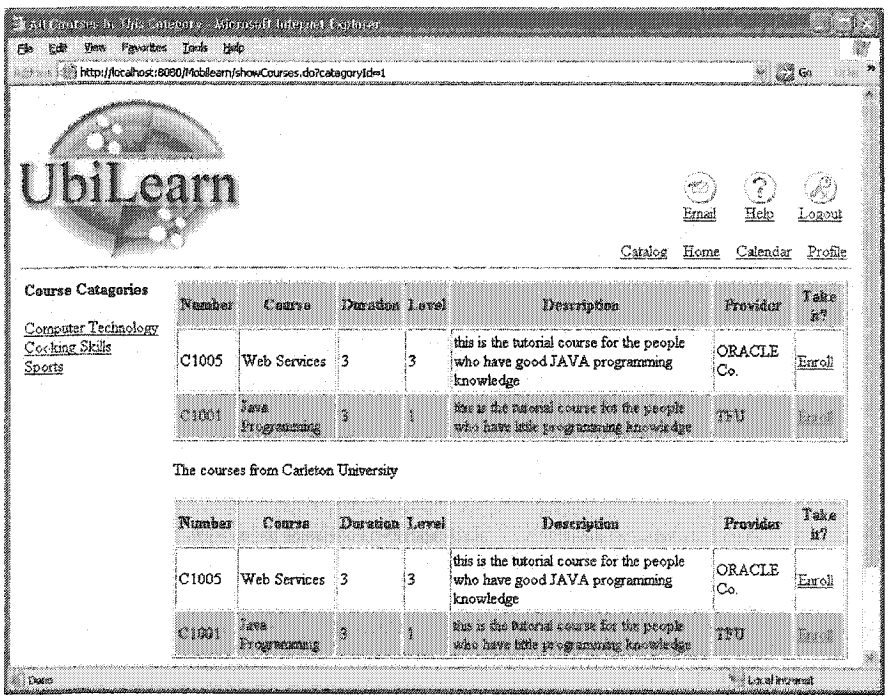



**Figure A-4** Local and Remote Courses for User Enrolment

## A.2 Main Interfaces for Wireless Devices


**Figure A-5** Main User Interfaces for Wireless Devices

# Appendix B: UbiLearn Installation & Startup[11]

## B.1 Needed Software

### B.1.1 Java(TM) 2 SDK SE

You must install the Java(TM) Software Development Kit Standard Edition. The SDK for win32, version 1.4.1 is on the Ubilearn setup CD (E:\Mobilearn_Setup\neededSW\ \neededSW\j2sdk-1_4_0_02-windows-i586.exe). For an operating system other than windows, or alternatively for a more recent version of the SDK, you down a free copy of the SDK from the Sun web site (http://java.sun.com/j2se/downloads.html)

#### B.1.1.1 Setup

- Run the j2sdk-1_4_0_02-windows-i586.exe file
- Follow the setup wizard instructions
- Note the installation directory. You will need it later
- Accept default setup options
- Make sure "java 2 runtime environment" is selected

### B.1.2 Java(TM) Web Services Developer Pack

You must install the Java(TM) Web Services Developer Pack. The Pack for win32, version 1.0.01 is on the Ubilearn setup CD (E:\Mobilearn_Setup\neededSW\neededSW\jwsdp-1_0_01-windows-i586.exe). For an operating system other than windows, or alternatively for a more recent version of the SDK, you down a free copy of the Pack from the Sun web site (http://java.sun.com/webservices/download.html)

#### B.1.2.1 Setup

- Run the jwsdp-1_0_01-windows-i586.exe file.
- Click next.
- Approve Sun's License Agreement.

*Finding the Java(tm) 2 SDK*

- When asked Select a Java(tm) 2 SDK, either select desired copy of the SDK from the list or click the "Browse" button to locate the SDK manually (If you have just

[11] This document takes the convention that the Ubilearn application and its entire list of needed software component will be installed on a windows machine on the C: drive and that the Ubilearn setup CD is located in the E: drive.

installed the SDK you will need to click the "Browse" button to locate the SDK manually).

- In the browsing window locate the Java 2 SDK SE installation directory. (If you followed default installation options installing Java2 SDK SE the folder will be c:\j2sdk1.4.0_02\.)
- Once the folder is located click "ok"

❖ When the desired copy of the SDK is listed and selected (radio box beside it is checked), click next.

*Http Proxy*

❖ If you need to setup http proxy information, click the checkbox and enter the information here. (Consult your network admin to know if you need the use http proxy or not and to obtain the information you need to enter here)

❖ If you don't need a proxy, simply click next.

*Tomcat User*

The Web Services pack comes with an Apache Tomcat web server. Tomcat needs an administrator

❖ Enter a user name and a password

❖ Note these, you will need them later

*JWSDP installation directory*

❖ On Windows XP the setup program will default the installation directory to "<home-directory>\jwsdp-1_0_01\". This can result in an inconveniently long installation path like this one "C:\Documents and Settings\Someone\jwsdp-1_0_01\".

❖ We strongly recommend you change this to shorter install path like "C:\jwsdp-1_0_01\". The rest of the installation document will assume that you use "C:\jwsdp-1_0_01\".

❖ Note the installation directory. You will need it later.

❖ Accept default setup options for the rest of the installation

### B.1.3 MySQL

You must install the My SQL Database server. MySQL Max version 3.23.53 for win32 is on the Ubilearn setup CD (E:\Mobilearn_Setup\neededSW\neededSW\mysql-max-3.23.53-win/setup.exe). For an operating system other than windows, or alternatively for

a more recent version of the server, you down a free copy of the MySQL from their web site (http://www.mysql.com/downloads/index.html)

### B.1.3.1 Setup

- Run the setup.exe file
- Follow the setup wizard instructions
- Note the installation directory. You will need it later
- Accept default setup options

*Generate Database Tables*

- Find the create-2.sql file on the UbiLearn setup CD (E:\Mobilearn_Setup\create-2.sql)
- Open a command prompt to c:\mysql\bin.
- Run "start mysqld-max-nt --standalone" command to start MySQL.
- Run "mysql" to go in the MySQL command window.
- At the prompt type "source e:\mobilearn_setup\create-2.sql"
- Don't pay attention to the loud beeps (they are ok)
- At the prompt type "use mobilearn"
- At the prompt type "show tables;"

```
+---------------------+
| Tables_in_mobilearn |
+---------------------+
| administrator       |
| ariadne             |
| country             |
| course              |
| coursecategory      |
| coursecontent       |
| dc                  |
| enrollment          |
| instructor          |
| learner             |
| months              |
| nameprefix          |
| provider            |
| state               |
| timezone            |
| user1               |
| usercategory        |
+---------------------+
17 rows in set (0.00 sec)
```

### B.1.4 J2ME Wireless Toolkit (J2MEWTK)

You must install the Java(TM) 2 Micro Edition Wireless Toolkit. The Toolkit for win32, version 1.0.4_01 is on the Ubilearn setup CD (E:\Mobilearn_Setup\wireless\ j2me_wireless_toolkit-1_0_4_01-bin-win.exe). For an operating system other than windows, or alternatively for a more recent version of the SDK, you down a free copy of the Pack from the Sun web site (http://java.sun.com/products/j2mewtoolkit/download.html )

#### B.1.4.1 Setup

- Run the j2me_wireless_toolkit-1_0_4_01-bin-win.exe file.
- Follow the setup wizard instructions.
- Install to "C:\ WTK104"

## B.2 Needed Libraries

The Ubilearn Application comes with all of its source code and compiled binaries. However, Ubilearn still needs the libraries for the MySQL database JDBC drivers and the source for the "Struts" application framework.

To make things easier, the UbiLearn setup CD contains a single folder with all the necessary files (for windows). If you already have these libraries on you system, you do not need to copy them.

For an operating system other than windows, or alternatively for a more recent version of these libraries, you can download a free copy of the Struts and/or MySQL JDBC drivers from their respective web sites (http://jakarta.apache.org/struts/acquiring.html) (http://www.mysql.com/downloads/api-jdbc.html)

### B.2.1 Copy Libraries

- Copy the source folder "E:\Mobilearn_Setup\Mobilearn" to its destination "C:\Mobilearn">

### B.2.2 Environment Variables

- In Windows XP open the environment variables window (right-click My Computer Icon , select Properties menu item, select Advanced Tab, click Environment Variables button)
- If you don't have a variable named "CLASSPATH" add one.

- Then add the following to the start of the variable: "c:\Mobilearn\struts\lib\struts.jar;c:\Mobilearn\mysql_jdbc;"

## B.3 UbiLearn Application Setup

The Ubilearn Application Binaries are accompanied with their source code. The binaries will work by them self's and to not need to be modified to make the application work. However, if you wish to modify the source code or if you are using a different Running Environment (Java(TM) 2 SDK, Java(TM) Web Services Dev Pack) than described in this installation document, you may need to recompile, and re-deploy the application using Ant (discussed in "UbiLearn Startup" Section on page 115).

### B.3.1 Copying the Binaries & Source Code to JWSDP

For the Java(TM) Web Services Developer Pack (JWSDP) to run the Ubilearn application you must copy the binaries to JWSDP.

- Under the JWSDP installation directory, find a folder named "webapps".
- On the UbiLearn setup CD (E:\Mobilearn_Setup\src-code), a folder named "Mobilearn".
- Copy the "Mobilearn" folder inside the "webapps" folder.

### B.3.2 Configure JWSDP to run the Mobilearn Application

For the Java(TM) Web Services Developer Pack (JWSDP) to run the Ubilearn application you must configure JWSDP to know where and how to run the Application.

#### B.3.2.1 Build properties

- Under .\webapps\Mobilearn\ find the build.properties files
- On a Windows XP machine your <home> directory is something like "C:\Documents and Settings\UserName\". On a UNIX machine your <home> directory is something like "/home/username/". In any case find your <home> directory.
- Copy the "build.properties" file to you <home> directory.

#### B.3.2.2 Environment Variables

- Under E:\Mobilearn_Setup, find the "environmentvar.txt" file and open it with notepad.
- In Windows XP open the environment variables window (right-click My Computer Icon, select Properties menu item, select Advanced Tab, click Environment Variables button)

- For each entry in the text file, add an entry or append to an existing entry in the system environment variables tables.

### B.3.3 Register the Application in the UDDI Registry

- In the start menu go to \Programs\Java(TM) Web Services Developer Pack and Click on "Start Xindice"

  Or, from the command prompt…

- Type "C:\j2sdk1.4.0_02\jre\bin\javaw.exe -classpath C:\jwsdp-1_0_01\bin com.sun.jwsdp.launcher.XindiceExecutable start"
- in file learningRegistry.properties (src and lib)
- change the first 3 IP for the UDDI
- the first IP is you own (in the lab 137.122.109.117)
- #2,3 → UDDI
- Change the "service name" in the files to some name of your choosing.

Then

- Run DOS cmd
- Go to UbiLearn folder
- Run "ant run-jaxr-publish"
- To see if it worked open the JAXR Registry Browser
- Fill the IP of UDDI
- Search by name

### B.3.4 Copying the Wireless Binaries & Source Code to J2MEWTK

For the J2ME Wireless Toolkit to run the Ubilearn wireless application you must copy the binaries to J2MEWTK.

- Under the WTK104 installation directory, find a folder named "apps".
- On the UbiLearn setup CD (E:\Mobilearn_Setup\wireless), a folder named "Mobilearn".
- Copy the "Mobilearn" folder inside the "apps" folder.

## B.4 UbiLearn Application Startup

Once you've performed all the steps in the UbiLearn Installation, your application should be installed. Those steps in the installation should only have to be done once.

However, to run the application, there are several components that must be started each time you start the Ubilearn application. These steps for each component will have to be repeated every time you stop any particular component and they must 'all' be repeated if you reboot the computer.

### B.4.1 Start TOMCAT

- In the start menu go to \Programs\Java(TM) Web Services Developer Pack
- Click on "Start Tomcat"

  Or, from the command prompt…
- Type "C:\j2sdk1.4.0_02\jre\bin\javaw.exe -classpath C:\jwsdp-1_0_01\bin com.sun.jwsdp.launcher.TomcatExecutable start"

### B.4.2 Register Application in TOMCAT

- Run DOS cmd
- Go to UbiLearn folder
- Run "ant install"

### B.4.3 Start the UDDI Registry (Only if UDDI is on your machine)

- In the start menu go to \Programs\Java(TM) Web Services Developer Pack
- Click on "Start Xindice"

  Or, from the command prompt…
- Type "C:\j2sdk1.4.0_02\jre\bin\javaw.exe -classpath C:\jwsdp-1_0_01\bin com.sun.jwsdp.launcher.XindiceExecutable start"

### B.4.4 Register application in UDDI (Only if UDDI is on your machine)

- Only once – you should only register the application in the UDDI registry if you have not already done so as part of the UbiLearn Installation.

### B.4.5 Start MySQL Database

- Open a command prompt to c:\mysql\bin.
- Run "start mysqld-max-nt --standalone" command to start MySQL.

### B.4.6 Start Wireless Application

- Click menu "KToolbar" to start J2MEWTK.
- Click "Open Project".
- Choose "Mobilearn".
- Click "Build" only if you have updated source code and want to rebuild it.

❖ Click "Run" to launch the application.

## B.5 Recompile with Ant - (Optional reading)

The UbiLearn Setup CD includes Application Binaries and Application Source Code. The binaries will work by them self's and to not need to be modified to make the application work. They have already been compiled against the environment descried in this installation document.

However, if you have chosen to use a different running environment and believe that you need to recompile the application or if you wish to modify the source code, you may recompile, and re-deploy the application using Ant.

❖ Ant needs to be installed (is it already with J2EE or JWSDP)

❖ Ant uses build files (build.xml)

❖ Run "ant build"

❖ Run "ant Install" every time you start Tomcat

Be careful of *.properties file in the source folders. They will overwrite those in lib folders.