CLEI Electronic Journal 5 (2), 2002

AVANTE: An architecture of CORBA components and XML Metadata for Web Based Instruction

Victor Theoktisto Adelaide Bianchini Edna Ruckhaus

Universidad Simon Bolivar
Departamento Computacion y Tecnologia de la Informacion
P.O. Box 89000, 1080A, Caracas, Venezuela

{vtheok, abianc, ruckhaus}@ldc.usb.ve

ABSTRACT

Most current solutions for Web Based Instruction (WBI) use a centralized management model and a proprietary internal representation. The AVANTE Architecture is a WBI environment assembled using CORBA distributed components, implementing core services such as course management, user authentication, collaborative work, database access, presentation, and others. The AVANTE components conform to a 4-tiered model, with Client, Presentation, Management, and Low-Level Services component sublayers. Emergent XML standards for WBI describe all metadata definitions. Components at the Management layer manipulate JDBC-SQL data from the Low-Level Services Layer, and combine it with corresponding XML Schemas, instantiating course objects as new XML descriptions and component services. A filter-mapping service in the Presentation layer produces the dynamic HTML web pages needed for user interaction, processing these XML descriptions by applying one or more previously defined XSL stylesheets. A similar mechanism implements interface customization and remote service administration. The developed WBI system was deployed with open source software. Adding CORBA components easily achieves on-demand scalability. Future services include auditing, adaptive interfaces, grading, content development, and integration with existing systems.

Keywords: Web Based Instruction, XML, XSL, CORBA, Distributed Components.

1 INTRODUCTION

Telecommunications, media, and computer technologies keep converging at an accelerated pace. Teaching has evolved greatly since the original face-to-face experience in Ancient Greece, evolving the correspondence schools of the mid-1800's, to last century's radio, television, VCR tapes, and satellite broadcasting. The application of the new Internet technologies in education and its implications are issues of vital importance for faculty, alumni, and society at large, where the need to keep up-to-date conspires against the sheer weight of generated knowledge and the decreasing time available for self-improvement. Many worldwide institutions are aware that *e–Learning* or "learning through the web" has exploded in the last few years, and are busily implementing changes in the traditional teaching/learning processes [28]. Even more so now that private companies have set up their own "corporate universities" by either buying or developing their own e–Learning solutions, and establishing associations with leading academic institutions.

Web Based Instruction (*WBI)* is a natural evolution of known modern methods to enhance teaching by incorporating computer software *(Computer-Assisted Instruction, Computer-Managed Instruction, Computer Based Education)*. It is higher level paradigm relying on the World Wide Web as the perennial repository of mutable instructional information, and Internet channels as the underlying content distribution systems [18]. A Web Based Instruction environment is defined as a set of instructional resources using the hypermedia attributes of the World Wide Web to create [15] a significant environment where learning is fostered and supported, applying a repertoire of cognitive instructional strategies within a constructivist environment that emphasizes collaborative work [22]. Collaborative Learning is the educational model underpinning WBI, defined as the set of instructional methods that motivate students to work together to obtain common academic objectives [13].

The AVANTE project started in September 1999 at the Universidad Simón Bolívar in Caracas, Venezuela (10000 students, 1200 faculty), with the main objective of developing WBI tools using emerging technology. Initially, a comparative study of current WBI environments [23] was carried out to contrast the strengths and weaknesses of available software tools, and how these tools could be used implementing different teaching methodologies. The project [25] is currently associated with the "*COMDIST: Distributed Components for Tele-presence Service Implementation*" program, with participating universities from Spain and Latin America and partially funded by CYTED-UNESCO.

During the next stage, the WBI tool's technical requirements were determined, which included software tools for instructional content design needing little or no programming experience, generic models allowing course delivery following a particular instructional approach, and support for collaborative work. According to these requirements, the architecture and interface design was completed. The development follows the Common Object Request Broker Architecture (CORBA) standard for distributed objects and Extended Markup Language (XML) templates describe all metadata components for architecture design, course interoperability and content interchange. Additionally, XSL stylesheets describe end-user interaction and presentation.

2 WBI DEVELOPMENT TOOLS

There are many WBI development environments currently available. Most have custom tools to ease the task of designing, running, and managing online courses, but require some knowledge of the Hypertext Markup Language (HTML). General features are shown in **Figure 1**.

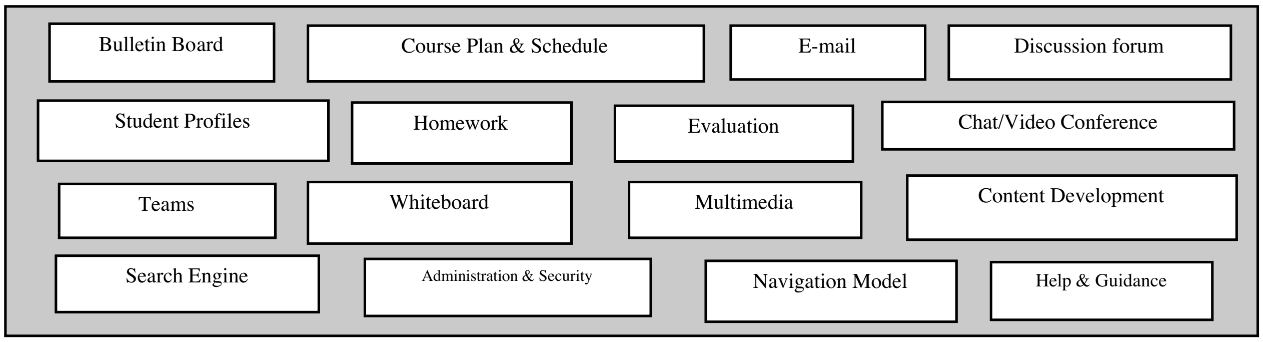


Figure 1: Features of a WBI environment

Some of these tools were developed by universities (i.e. WebCT), and others by software companies (i.e. Learning Space). Existing surveys compare features, technical requirements, and costs. A sampling of features, not uniformly present or required include the following: templates for course development, adaptive content and user interfaces according to student profile and performance; resource sharing; differentiated roles for course designers, instructors, administrators and users; a course administration hierarchies (course supervisor, instructor, assistants, student); and a language standard for metadata.

These initial requirements were the starting point for the following stage of AVANTE, which produced a more detailed version of the WBI tool's requirements and desirable functionality.

3 REQUIREMENTS FOR BUILDING WBI TOOLS

The following tasks were completed to determine the WBI tool's final requirements:

Analysis of the traditional teaching/learning process, its main activities and their sequence.

The main activities in course preparation and teaching: lectures, material distribution, answering student questions, group activities, homework, project supervision, and grading. The instructor uses several media resources to enrich content, motivate students and widen communication channels. The student annotates, goes over the material, asks the instructor for clarification, and gets feedback either directly or indirectly through individual evaluation. The instructor is an information emitter, the student is a receiver, and the effect of content delivery is measured through assignments and evaluations.

Definition of teaching/learning schemes with active student participation and instructors as facilitators.

The instructor promotes active learning experiences through collaboration, and therefore students develop group cognitive skills such as analysis, synthesis, critical thought, and knowledge application. Each class session becomes a student-centered forum for discussion. The students are ultimately responsible for their own knowledge construction.

Analysis of models supporting new technologies in education

Internet technologies promote communication among actors of the teaching/learning process, facilitating collaborative work, and distribution of up-to-date information.

According to [6] the four basic stages for incremental WWW use in learning are:

i) *The web as an information source*. Many instructors publish a homepage with static course information such as the syllabus, administrative details, schedules, course references, and lecture material.

ii) *The web as an electronic book*. The web is used to store structured course content, structured with exercises and examples. Students follow instructions and work in almost the same way they deal with printed media.

iii) *The web as a teacher*. Instructors communicate with students on synchronous and asynchronous channels, such as e-mail, bulletin boards, threaded discussion phora, chat rooms, and videoconferences.

iv) *The web as a vehicle*. The most elaborated model provides the instructor with the tools to create and present online structured content, with all necessary features for a virtual face-to-face interaction with students.

Definition of Teaching/Learning requirements

Since students interact with course content according to need, rhythm, and grouping, a WBI tool should be highly interactive [32]. A proper WBI should not be limited to merely browsing course materials.

Besides student access [27], a WBI tool must support the following actions:

- *Promote student-group interaction, generating different perspectives on every learning task.*
- *Promote student-instructor interaction.*
- *Provide students with peer-evaluation capabilities in order to exchange information and solve problems.*
- *Foster a collaborative environment.*

On the other hand, instructors must have the following possibilities:

- *Participate and mediate constantly in learning activities.*

- *Supervise student assignments.*
- *Motivate student teams to carry out their assignments.*
- *Check progress of assignment's activities.*
- *Answer requests and questions from students.*

Summing up, the following requirements were set up for AVANTE:

i) *The users are students, instructors, and system administrators. Instructors do not need to have any knowledge of programming techniques to create web material. Students may belong to any discipline.*

ii) *Simple creation of didactic material based on instructional strategies.*

iii) *Transparent creation and maintenance of content databases, with fast information retrieval, and links to bibliographical databases.*

iv) *Platform independence. It must be flexible and open, allowing implementation of proprietary or open source solutions without impeding a transparent interaction by faculty and students alike.*

v) *A consistent, customizable, easy to use Multimedia User Interface.*

vi) *Interaction among system actors must allow several communication channels and modalities, for either information interchange or a mediated discussion.*

vii) *Scalability, easy maintenance.*

The following objective reflects these requirements:

"To develop a web-based environment for learning services, using synchronous and asynchronous channels, with a platform-independent tool set for course creation, administration, information retrieval, automated publication of instructional content, and user interface customization".

4 STANDARDS AND SPECIFICATIONS

Many technical difficulties limit e-Learning service standardization. Most sites rely on raw HTML and CGI services, and others are managed by applications [11] (perl, php, asp, jsp, cfm) serving HTML pages combining predefined page chunks with dynamically assembled data. However, an HTML page describes how data is presented, but does not include semantic or ontological tags. Administrators and instructors have a very difficult time evolving or simply maintaining their on–line courses. The non-portability of course content has spawned an abundance of derelict course websites.

Recently, several initiatives have arrived to unify metadata approaches for web-service development at all levels. To solve HTML's lack of semantic content, the *World Wide Web Consortium (W3C)*, the body in charge of specifying web standards, has developed the *Extensible Markup Language (XML,* an SGML-derived markup language for content tag definition [31], having *Document Type Definitions metadata (DTDs)* that check a document's semantic validity. XML is particularly well suited for representing hierarchical data. A DTD is an XML definition describing the logical structure of a particular class of documents. This allows the separation of the document's content from its layout, and software tools can automatically parse the document.

An *XML schema definition* (XSD) is a formal specification of element names establishing the allowed combinations in an XML document. Schema languages are more powerful than DTDs, and may be extended with data types, inheritance, and presentation rules.

XML documents are transformed into other document formats (HTML pages, additional XML documents) using *Extensible Style Language* templates (XSL). These templates handle presentation and feature extraction, where an input XML file defined under a particular XSD definition is transformed into an output document by applying an previously defined *XSL transformation* (XSLT) linked to the same schema.

A coordinating body known as the *Dublin Core Metadata Initiative* (DCMI) [9] promotes the widespread adoption of interoperable metadata standards to avoid proliferation of XML specifications, setting up *namespaces* (specialized metadata vocabularies) to describe resources and enable intelligent information discovery systems. The actual XML education namespace (*DC-ed*) is a work in progress [8].

The service architecture for the specification of learning objects (**Figure 2**) is coordinated by the *IEEE Learning Technology Standardization Committee* (IEEE LTSC*)*. The current standard is the *Learning Object Metadata (LOM)* draft [17], based on DCMI. The IEEE has formed several committees, the most important being ARIADNE and IMS [14], in charge of the LOM model kernel. Since 1996, the European Union has sponsored the ARIADNE Project *(Alliance of Remote Instructional Authoring and Distribution Network for Europe)*, focusing on the development of software tools and methodologies for producing, managing and reusing computer-based pedagogical elements in a internet-supported training curricula [1]. Validation of the project's concepts is currently taking place in various academic and corporate sites across Europe.

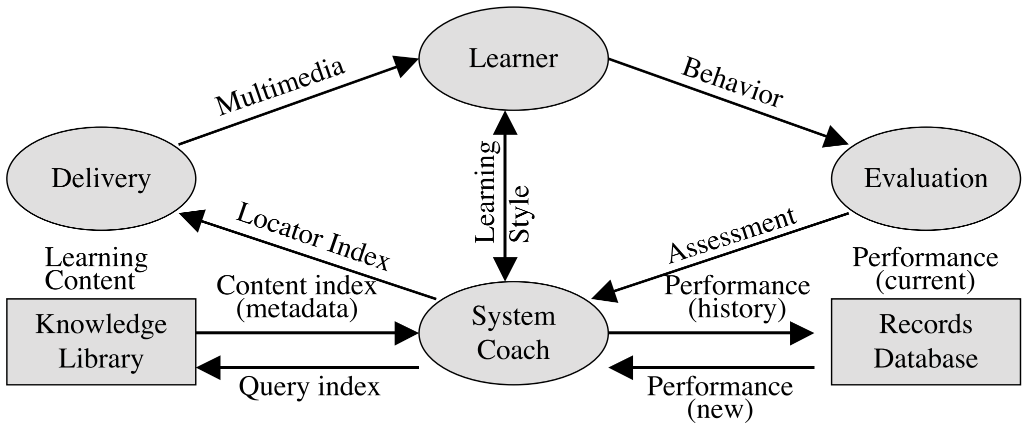


Figure 2: IEEE LTSC Service Architecture

The IEEE LOM draft defines nine meaningful categories of XML descriptors for learning objects, with a more detailed subcategory description in [17]:

1. General: *context-independent features of the resource, i.e. Identifier, Title or Human Language.*
2. Lifecycle*: features related to the life cycle of the resource, i.e. Version or Status.*
3. MetaMetaData: *origin and edition of the metadata.*
4. Technical: *technical features of the resource, i.e. Format (technical data type of the resource).*
5. Educational: *educational or pedagogic features of the resource:*
6. Rights Management: *features that need to be interpreted according to resource use.*
7. Relation: *features of the resource in relationship to other resources.*
8. Annotation: *comments on the educational use of the resource.*
9. Classification: *describes where this learning object falls within a particular classification system.*

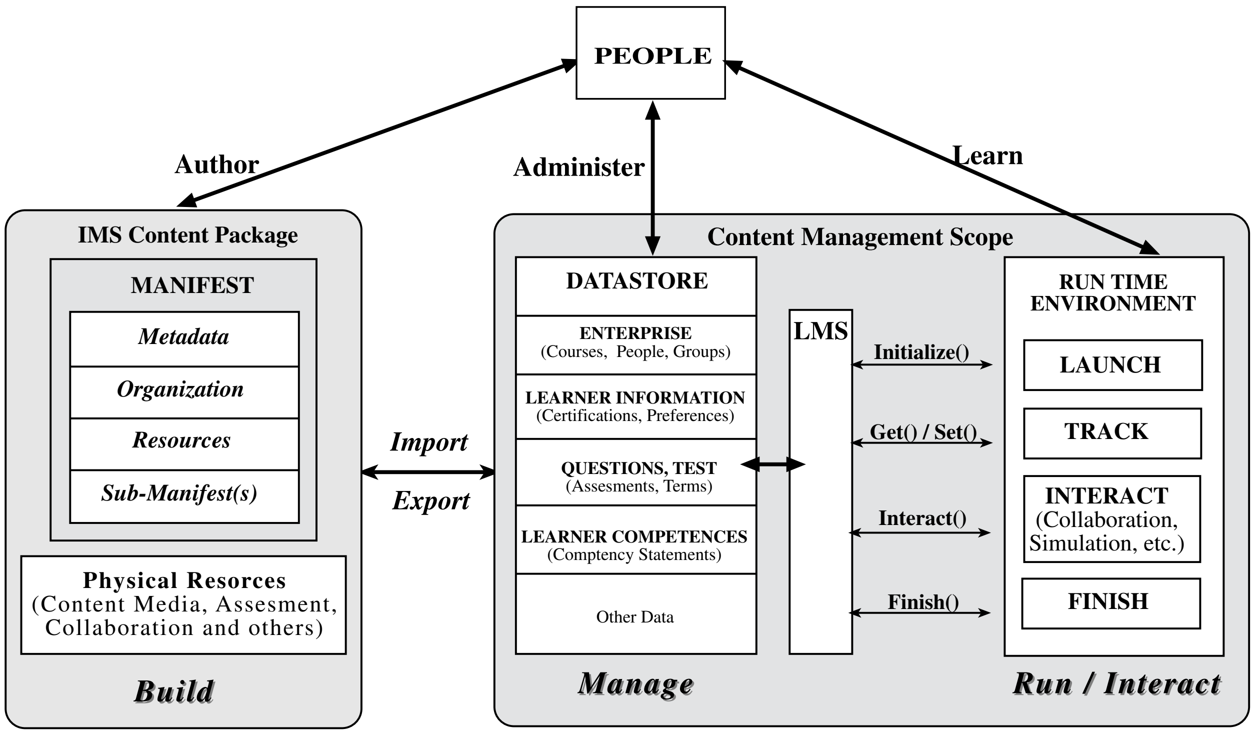


Figure 3: IMS Content Framework

Content and Metadata XML specifications

Based on the LOM model, the *Instructional Management Systems Project (IMS)* has developed XML specifications for course metadata information.

The *IMS Framework* shown in **Figure 3** describes the *IMS Content Package* and its two major elements: (i) the *IMS Manifest,* XML files describing content organization and resources in a package; and (ii) the physical files being described by that XML. A package that has been incorporated into a single file for transportation (e.g., .zip, .jar, .cab) is called a Package Interchange File.

The CORBA Specification

The *Common Object Request Broker Architecture (CORBA)* is regulated by the *Object Management Group (OMG)* [19], and backed by the major software and hardware companies. It has the objective of promoting object-orientation in software engineering. The OMG [7] has adopted interfaces and protocols on commercially available objects, defining a bus architecture for object organization, which is an extension to the Client/Server model. This is accomplished by specifying a common standard for the development of distributed applications, which relies on the reusability, portability, and interoperability of objects in heterogeneous environments.

The CORBA bus defines the components' structure and the manner of interaction, allowing low-level communication among objects by synchronized remote procedure calls (RPCs). CORBA also allows asynchronous interaction, so clients can continue working after issuing a request without having to wait an immediate result from the server. CORBA allows classes to be implemented in several languages, executed on different operating systems and platforms, and dispersed on a heterogeneous network. To allow this, CORBA centers on three main concepts:

Separation between interface and implementation. All CORBA components are specified by an Interface Definition Language (IDL). IDL is a purely declarative language close to C++, but without statements or control structures. It is implementation independent, with existing bindings for C, C++, Java, Ada, Smalltalk, and others [3]. A component can specify which classes it inherits from, method signatures, attributes, "throwable„ exceptions, input and output arguments, return values, and data types.

Localization Independence. The kernel of any CORBA implementation is the *Object Request Broker (ORB),* which is a naming service that makes the localization of objects transparent. It routes petitions in such a way so that objects can communicate among themselves, whether on the same machine or across a network.

Vendor Interoperability and Systems Integration. The *Internet Inter-Operability Protocol (IIOP)* specifies how to change the messages from *General Inter-ORB Protocol (GIOP)* in TCP/IP networks, making the Internet a giant ORB, where other ORBs interact without problem as shown on **Figure 4**. The IIOP has the advantage of operating on the *Secure Sockets Layer (SSL)*, which allows secure (encrypted) passage of data through the bus.

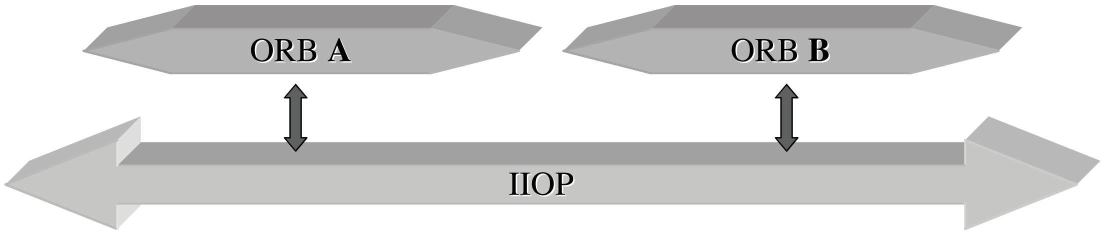


Figure 4: IIOP Structure

The architecture allows the creation of simple objects that may have transactional, secure, locked, or persistent attributes when inheriting from the proper services. These objects are able to interrelate and find themselves within the bus. CORBA specifies an extensive service set for the creation, disposal, name access, storage, and retrieval of objects, and how to define relations among them.

All CORBA Services are implemented in IDL (**Figure 5**). The ORB's interface offers dynamic invocation interfaces to CORBA objects and the *Stubs* service for static ones. Operations allow translation of object references as character strings, with methods for handling null verification, duplication, and so on. It also has reflexive metainformation functions that interrogate objects about their type. From the client side, to invoke a remote object with a local reference, the former must be referenced first, and then the client is linked with an IDL-defined *Stub*, transforming the ORB's request into a real implementation that inherits the interface methods. On the server side, for each interface there is an implemented *Skeleton*, transforming the requests from the ORB into object server invocations. The IDL compiler generates both *Stubs* and *Skeletons*.

To avoid the linking of all clients with all possible *Stubs* at the server side (static invocation), a client may perform dynamic invocations of objects operations for which it does not have proper *Stubs* through the *Dynamic Invocation Interface* (DII). All this allows CORBA to work in dynamic realms on which new components are added at execution time. The 2.3 standard defines the *Portable Object Adapter* (POA), which manages the creation, registry, destruction, reference handling, and activation of the implemented objects. There is a definable set of policies that specifies POA's behavior on its associate objects

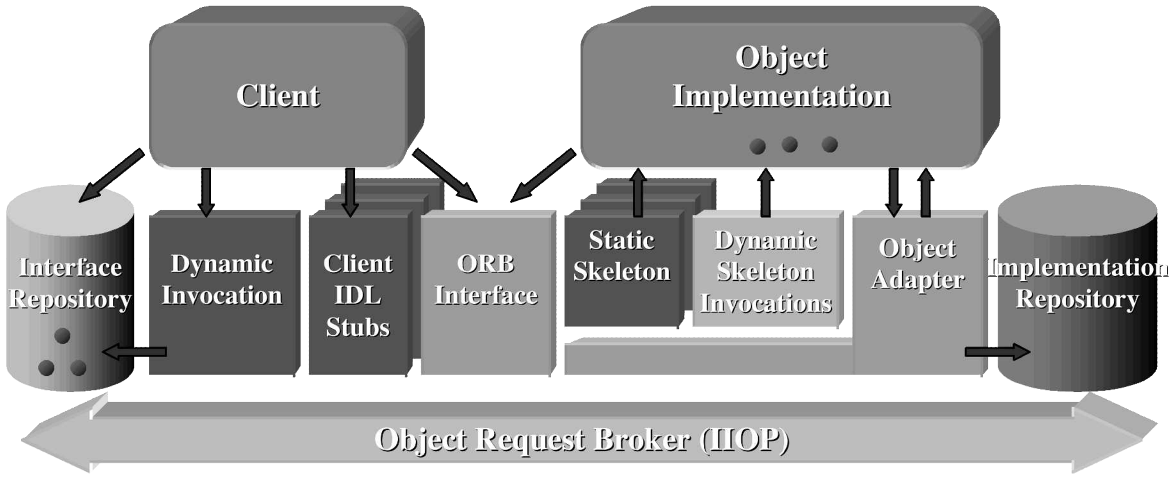


Figure 5. The CORBA Service Architecture

CORBA Facilities and Services

The standard service set established by the OMG is comprehensive and growing:

a. *Naming Service:* Object registry, component name lookup, and hierarchical context ("folders„).
b. *Event Service.* Dynamic decentralized event registry for components.
c. *Life Cycle Service.* Defines create, copy, move and kill operations for bus components.
d. *Persistence Service.* Provides a unique interface for alternative component Databases.
e. *Relationship Service.* Handles dynamic links among components.
f. *Externalization Service.* Extracts or inserts data in a component using *streams*.
g. *Transaction Service.* Provides a two phase dedicated coordination between components with atomic transaction control.
h. *Concurrency Control Service.* Provides a locking system for *threaded* synchronization.
i. *Licensing Service.* Measures components used to compute just compensation.
j. *Query Service.* SQL-based object queries.
k. *Properties Service.* Assigns values or properties to component objects such as dates.
L. *Security Service.* Provides a secure working environment for distributed objects.
m. *Time Service.* Provides interfaces for time synchronization of distributed objects.
n. *Collection Service.* Provides CORBA interfaces for handling of the most common collections.
o. *Trader Service.* A Yellow Pages service so distributed objects promote themselves.

This gives an idea of power of CORBA object to help implement distributed services on the web [4], [5].

4 THE AVANTE ARCHITECTURE

The AVANTE project (Aula Virtual – Aulas de Nueva Tecnología) started in September 2000 as an improved alternative for WBI. It is platform independent development being using open source software for a standards-based design. The Linux OS platform was chosen due to its near universal availability for different processor architectures (Intel, PowerPC, MIPS, SPARC), its robustness as the major server environment, and its growing code base.

There are three physical layers in AVANTE, which are shown in **Figure 6**. The first layer is an Apache™ web server array that handles incoming http requests. A servlet engine is running on each server (Tomcat™ or Resin™) [10] [12], in parallel with a Java Virtual Machine [30]. This allows the session control to be closer to clients, and hides the CORBA interaction, with almost no performance penalty.

The second layer is the Component layer, which hosts all CORBA services. The last layer is the Database repository, which is accessed with JDBC-based communications. It may be any JDBC-SQL compliant database environment [24], such as ORACLE, Postgres, or similar. All secure communication is maintained through SSL.

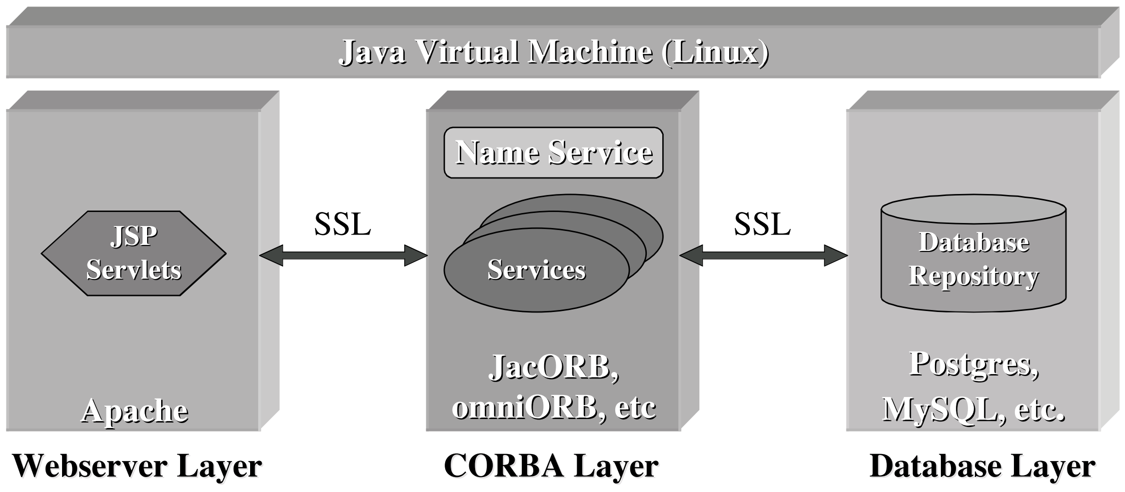


Figure 6: AVANTE Physical Layers

The CORBA component layers

All WBI functionality is implemented as CORBA distributed services. There are four sublayers in the CORBA layer, with services grouped according to their nature [26]: Client Services [20] (which right now are handled by the servlet engine), Presentation Services, Management and Low Level Services layers, as shown on **Figure 7.**

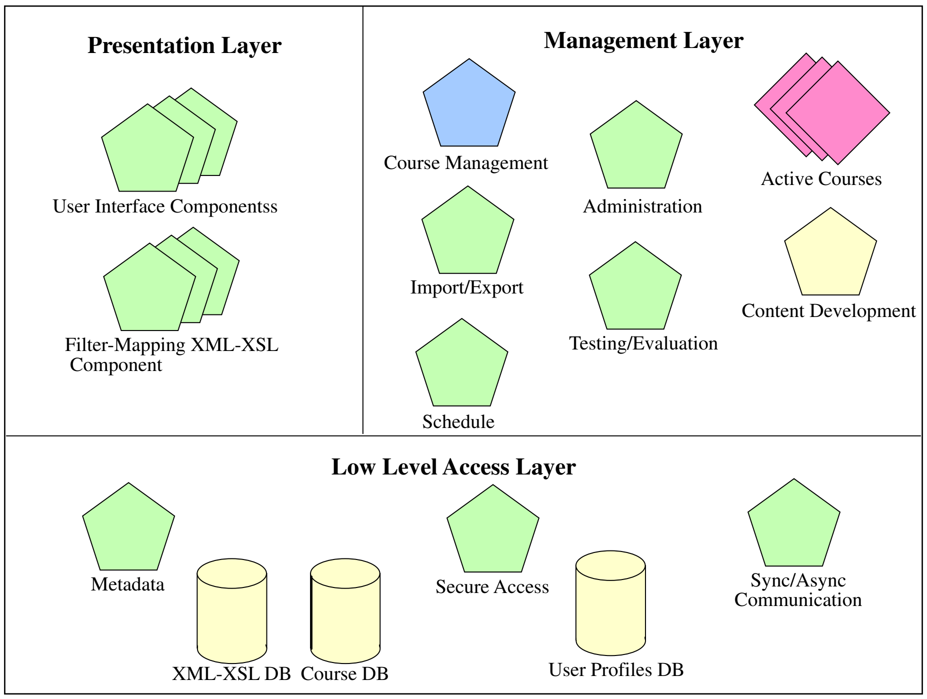


Figure 7: AVANTE Component sublayers

All CORBA services inherit from the same classes, shown using UML notation in **Figure 8** [16] [21] [29]**.**

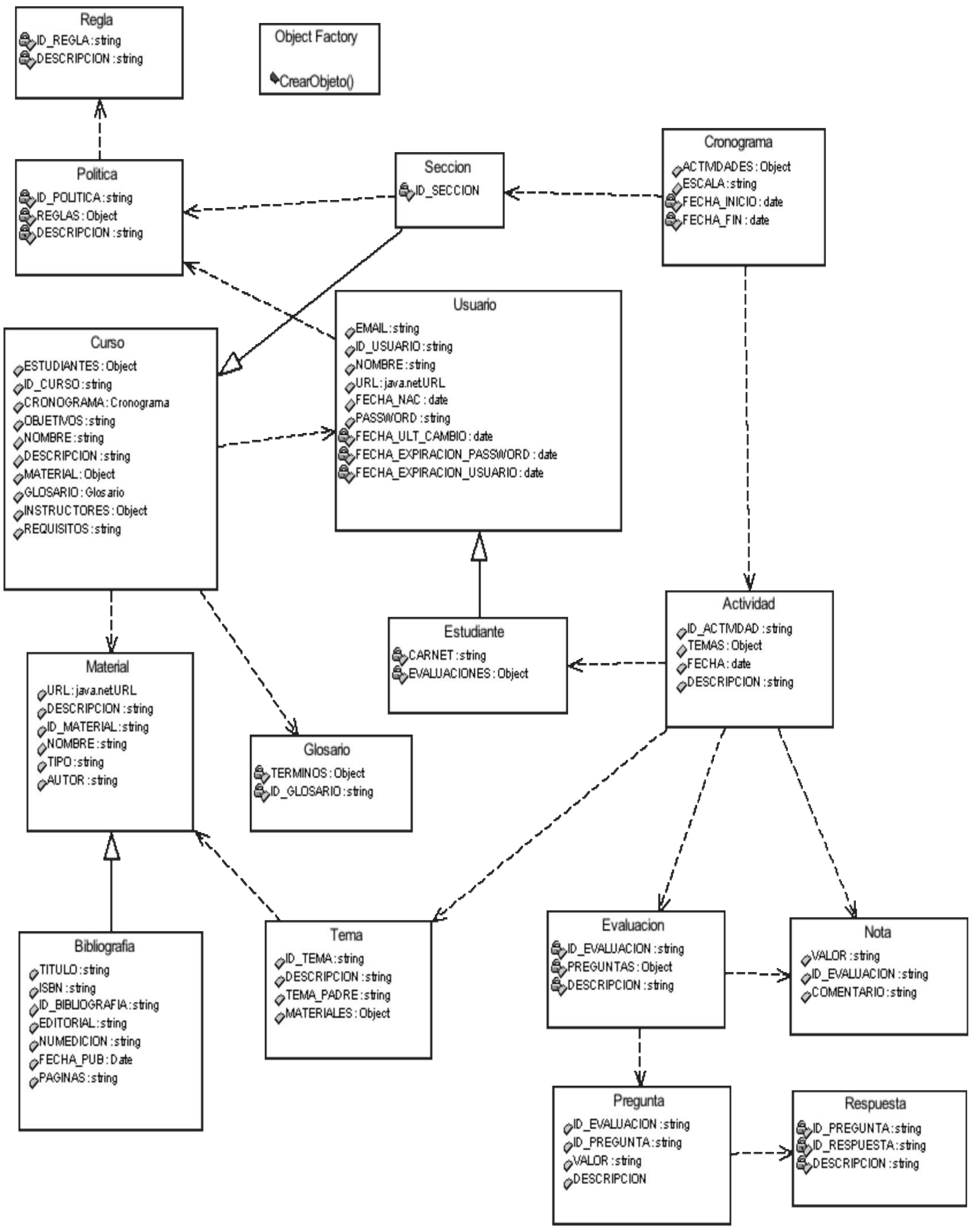


Figure 8: Course Component Class Diagram

A Typical Session with AVANTE

An AVANTE session may started using any web browser. The responsibilities of all CORBA components are shown in Table 1. Five main user classes have been defined: Creator, Instructor, Student, Browser, and Administrator, which can be further customized to special course needs.

A Creator user defines syllabus, content, and methodology. An Instructor uses an already defined course, and activates it, adding class schedule, evaluations, grade reporting, and communication channels. A Student interacts with all the course facilities, and may be allowed to add material. The Browser or Guest has very few rights, and the Administrator preloads courses with student records, class enrollment lists, and takes care of all necessary security and performance enhancement.

When a course is created and later instantiated, a CORBA object is created as a separate service representing the course, and from then on, it exists with persistent and serializable properties. Students, after successfully logging and obtaining their profiles, find a course using the CORBA name service, which activates the course by the servlet running on the web server.

Table 1: AVANTE Component Description

Layer	Component	Description
Management	Course Management	*Course Design and Maintenance*
	Active Courses	*Active Course Objects*
	Administration	*Environment Configuration*
	Auditing	*Logging and Tracking statistics*
	Syllabus/ Class Schedule	*Class Planning*
	Testing/Evaluation	
	Grade Reporting	
	Content Creation	*Bridge for content creation tools*
	Tutoring Methodology	*Teaching/Learning strategies*
Presentation	User Interface	*Interaction*
	User Profiles	*User Customization/History*
	XML-XSL Filter-Mapping	*XSL-XML translator*
Low Level Services	Access/Session	*Secure access and user authentication*
	Synchronous and asynchronous Communication	*e-mail, chat, news, whiteboard, videoconference channels*
	Import/Export Packaging	*Data and metadata migration*
	Metadata Handling	*XML creation and storage*
	Database Access	*JDBC-SQL connection pool management*

The XML-XSL Translator Component

Course and student information are pulled from the corresponding database, and converted to XML metadata representation. The XML-XSL Filter-Mapping Component transforms this representation using associated XSL templates, and producing the instantiated data needed for the next processes. As may be appreciated in **Figure 9,** this is applied to create customized courses and student web pages. The Filter-Mapping Component is also used for course package conversion to transport formats.

Coding filter components is straightforward, and most of them consist of the following three lines of code:

```
get reference
obtain new_reference by applying transformation to reference
return new_reference
```

XML-XSL filters are also used at the Database Component layer to produce XML data and objects. This allows a simple yet powerful way of exchanging XML data with SQL databases, as shown in **Figure 10**. A filter matches XML metadata with some corresponding XSL style sheet having all the necessary SQL statements, generating an SQL query, and the results merged and formatted as described by the XML template. Different queries may be expressed using appropriate XSL styles, resulting in HTML files or other XML templates.

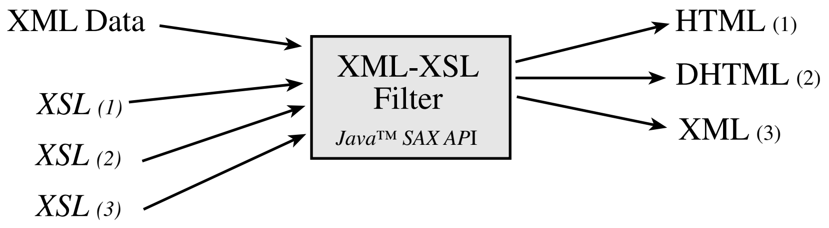


Figure 9: XML-XSL Mapping

The Database Access Component

XML-XSL filters are also used at the Database Component layer to produce XML data and objects. This allows a simple yet powerful way of exchanging XML data with SQL databases. As shown in **Figure 10**, a filter matches XML metadata with some corresponding XSL style sheet having all the necessary SQL statements. A query is generated, and the results merged and formatted as described by the XML template. Different queries may be expressed using the appropriate XSL's.

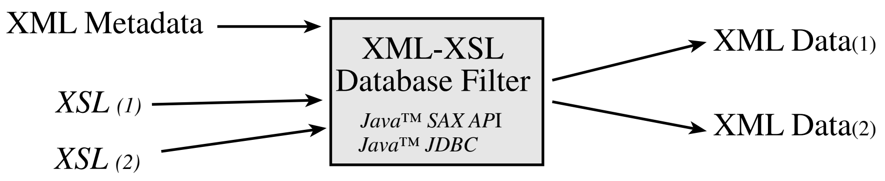


Figure 10: XML-XSL Database Mapping

5 A SHORT EXAMPLE

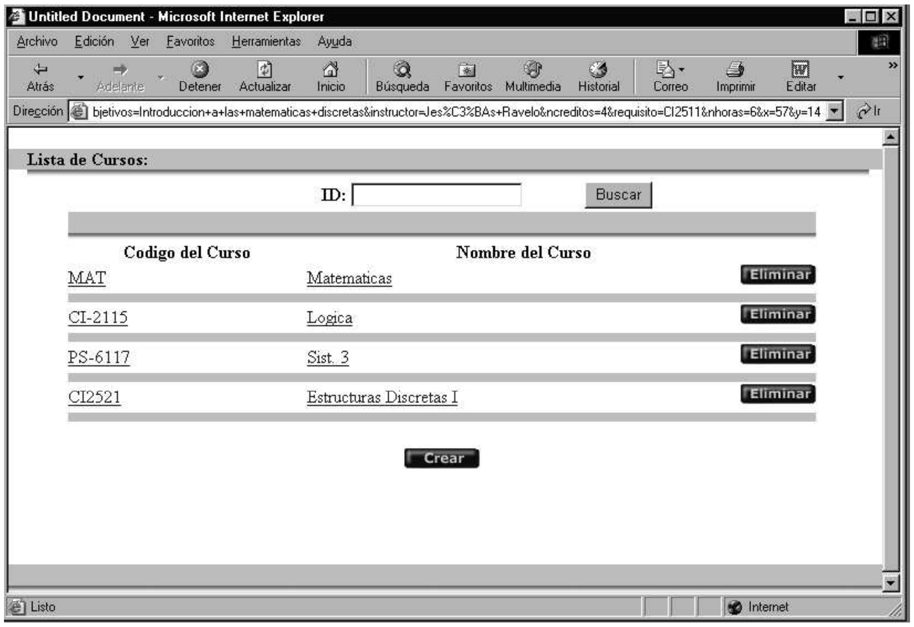
The following example shows how to build a webpage that offers access to a list of courses stored in a database using an XML metadata description:

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href= "CourseCreation.xsl" ?>
 <COURSE>
   <COURSE-NAME></COURSE-NAME>
   <COURSE-CODE></COURSE-COD>
   <DESCRIPTION></DESCRIPTION>
   <REQUISITES></REQUISITES>
   <NHOURS></NHOURS>
   <NCREDITS></NCREDITS>
…
    <SECTIONS>
     <ID>
    </ID>
    </ SECTIONS >
 </COURSE >
```

A database access component uses the Java Xerxes XML parser API and the XML:DB API to traverse an XML, build a query, pull the necessary records and fields from the database, and finally output an instantiated XML file with the query's result.

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href= "CoursePresentation.xsl" ?>
 <COURSE>
   <COURSE-NAME>Estructuras Discretas</COURSE-NAME>
   <COURSE-CODE>CI-2521</COURSE-COD>
   <DESCRIPTION>Some description required</DESCRIPTION>
   …
</COURSE >
```

The same procedure applies to each course that is to be displayed. Using the presentation XSL, a webpage is built on-the-fly and presented to the user. All the user interface elements (buttons, images, etc.) are specified in the XSL template. The final page is served by a JSP server using the Xalan and Xerxes APIs. The two filter components implementing this process are so generic in nature that become a design pattern for distributed information services.


Untitled Document - Microsoft Internet Explorer
Archivo Edición Ver Favoritos Herramientas Ayuda
Atrás Adelante Detener Actualizar Inicio Búsqueda Favoritos Multimedia Historial Correo Imprimir Editar
Dirección
Ir
Buscar
Eliminar
Crear
Listo
Internet

bjetivos=Introduccion+a+las+matematicas+discretas&instructor=Jes%C3%BAs+Ravelo&ncreditos=4&requisito=CI2511&nhoras=6&x=57&y=14

Lista de Cursos:

ID:

Codigo del Curso	Nombre del Curso
MAT	Matematicas
CI-2115	Logica
PS-6117	Sist. 3
CI2521	Estructuras Discretas I

6 CONCLUSIONS

Security issues inherent to the CORBA model must be solved for universal unhindered access. In this version no provision was made for firewalls, only those provided by the secure socket layer. An easy solution requires that users, ISPs, and administrators know and authorize the specific ports on which to connect. Another alternative would be to funnel all connections through HTTP tunneling. The chosen servlet solution in AVANTE avoids both problems, leaving all CORBA connections to the servlet maintaining user sessions.

The AVANTE Architecture was designed with extensibility and maintainability in mind. Based on available open standards, it allows the packaging of courses and future migration to compliant WBI tools. By using open source software readily available on the web [Linux™ OS, Apache™ Web Server, Resin™ Servlet Engine, the Java Virtual Machine™, Xerxes, Xalan, XML:DB], the environment can be set up and deployed at no added cost. For solutions requiring better performance, proprietary solutions that conform to open standards may be used instead, such as ORACLE for database access, or a more complete object request broker such as ORBacus or ORBIX. Users even text-based ones, since no processing gets done at the local level. However, some web browsers can now render a page by locally processing XSL and XML, so this may change in the future.

The underlying reliance on XML models and the use of XSL-based translation services assures that valid changes in data and metadata do not affect existing components. There is precious little code and most of it is of generic nature. System behavior and presentation may be changed just by modifying the XML specifications and XSL transformations. Existing services can be added on demand, providing a framework where new services are integrated very easily. The CORBA component model implemented allows for on-demand scalability, since replicated services may be spread as new server machines are added. New services will be added to enhance user experience and system capabilities. These include event logging and auditing, dynamic adaptive user interfaces, on-line testing/evaluation, and integration with existing administrative systems.

Further work is related to the development of the content creation component, and a model for instructional design [2] has been developed in which the instructor will be able to select learning objectives and strategies to attain them, including all the needed elements for content design and related learning activities. The AVANTE Architecture provides a blueprint for standards-based development of distributed services, with can be successfully applied to almost to other application domains in distributed objects.

REFERENCES

1. ARIADNE Project: Alliance of Remote Instructional Authoring and Distribution Network for Europe, est. 1996. http://ariadne.unil.ch/Metadata/ariadne_metadata_v3final1.htm

2. Bianchini, A., "Modelo referencial de hipermedio basado en teoria de grafos para minimizar el problema de desorientacion del usuario". Proceedings CIC'2000. Pp. 121-130. Mexico. November 2000.

3. Brose, G., "Reflection in CORBA, Java and JacORB". Institut für Informatik Freie Universität. Berlin, 1998

4. Brose, G., Noffke, N., and S. Muller., "Jacorb 1.2 Programming Guide". Institut fur Informatik. Freie Universitat, Berlin 2000

5. Brose, G., Vogel, A., and K. Duddy, "Java Programming with CORBA", 3rd Ed. John Wiley & Sons, New York, 2001.

6. Casey, D. "Learning from or through the Web: Models of Web-based Education." SIGCSE-Bulletin 30 n. 3, pp. 51-54, 1998

7. Curtis, D., "Java, RMI and CORBA". Object Management Group. http://www.omg.org/news/wpjava.htm. http://dublincore.org/documents/education-namespace/

8. Dublin Core Namespace Proposal, http://dublincore.org/documents/education-namespace/

9. Dublin Core Metadata Initiative, 1995. http://dublincore.org/

10. Ferguson, S., "Resin Overview". Caucho Technology, California, USA, 2001 http://www.caucho.com/articles/resin_overview.xt

11. Garshol. M., "What's Wrong with Perl" (1998). http://www.stud.ifi.uio.no/~larsga/download/arti-kler/perl.html.

12. Hall, M., "Core Servlets & Java Server Pages". Sun Microsystems Press, USA, 1998.

13. Hiltz, R.: "Teaching in a Virtual Classroom". International Conference on Computer Assisted Instruction, 1995.

14. IMS Content Packaging Information Model v1.1. IMS Global Learning Consortium, Inc. April 19, 2001

15. Khan, B., “Web-Based Instruction (WBI): What is it and why is it?” In B.H. Khan (Ed.), Web-based Instruction. Englewood Cliffs, NJ: Educational Technology Publications, Inc., 1997.

16. Larman, C., “UML y Patrones”. Prentice Hall, México., 1998.

17. Learning Objects Metadata, Rev. 6. IEEE P1484.12/D6. IEEE Learning Technology Standardization Committee. February 2001

18. Mathew, N., and M. Dohery-Poirier, “Using the World Wide Web to Enhance Classroom Instruction”. First Monday. Peer-Reviewed Journal on the Internet. Vol. 5., N. 3. March 2000. http://www.firstmonday.org/issues/issu5_3/mathew/index.html

19. Object Management Group “The Common Object Request Broker: Architecture and Specification (CORBA 2.3)”. USA, 1999

20. Harkey, D., and R. Orfali, “Client Server Programming with Java and CORBA, 2nd. Ed”. Wiley Computer Publishing, New York, USA, 1998..

21. Rational Software Corporation. “RUP White Paper”. USA, 1999.

22. Relan, A., and B. Gillani, "Web Based Information and the Traditional Classroom: Similarities and Differences”. In B.H. Khan (Ed.), Web-based Instruction. Englewood Cliffs, NJ: Educational Technology Publications, Inc., 1997.

23. Ruckhaus, E., and V. Theoktisto. "Aula Virtual I: Informe Proyecto DID S1-CA-411”. USB, Caracas, 2000.

24. Sun Microsystems Inc. “JDBC 2.0 API”. Palo Alto, USA, 1998

25. Theoktisto, V., Bianchini A., and E. Ruckhaus, “Aula Virtual II: Extension al proyecto DID S1-CA-411”. USB, Asociado al proyecto CYTED-UNESCO. Caracas, 2000.

26. Theoktisto, V., “Patrones de Diseño”. Universidad Simon Bolivar, Caracas, 2000

27. Turgeon, A., "Web-based instruction with modules for active learning." ADEC Presentation. September 1999. Url: http://www.adec.edu/workshops/1999/sept28/outline.html

28. Ubell, R., “Engineers Turn to e-Learning”. IEEE Spectrum, October 2000.

29. Rational Software Corporation, “UML Notation Guide”. USA, 1998.

30. Vogel, A., and K. Duddy, “Java Programming with CORBA, 2nd. Ed.” Wiley Computer Publishing, New York, 1998.

31. Extensible Markup Language (XML) 1.0 (2nd edition). World Wide Web Consortium, 6 October 2000. http://www.w3c.org.

32. Yueh-Chun Shih & Nian-Shing Chen: “The Conceptual Model of Web-Based Instruction System and Its Implementation”.ICCE 99. http://www.nsysu.edu.tw/~nschen/conference/icce99.htm