Silva, V., Lucena, C.J.P. & Fuks, H. (2001) ContentNet: A Framework for the Interoperability of Educational Content Using Standard IMS, Computers & Education Journal, Vol 37/3-4, Elsevier Science Press, ISSN 0360-1315, pp. 273-295

# CONTENTNET: A FRAMEWORK FOR THE INTEROPERABILITY OF EDUCATIONAL CONTENT USING STANDARD IMS

Viviane Torres da Silva    Carlos José Pereira de Lucena    Hugo Fuks

Software Engineering Laboratory - Computer Science Department
Pontifical Catholic University of Rio de Janeiro – PUC-Rio
Rua Marquês de São Vicente, 225 - Gávea
Rio de Janeiro - Brazil
Phone/FAX: + 55 (21) 512-2299
{viviane; lucena; hugo}@inf.puc-rio.br

## Abstract

The World Wide Web arises as a means to facilitate communication and provide an easier user interface to obtain distributed data. The WWW was organized to allow information contained within it to be read by machines, without them needing to be concerned about interpreting the content. With the increase in the volume of information on the Web, finding, accessing and obtaining information from it became extremely difficult.

Seeking a solution to this problem, the IMS project, a consortium of academic, commercial and governmental organizations, has been developing and proposing specifications to facilitate the growth and viability of online activities in the area of education. One of the most important activities of the project is the learning process on the Internet, with the possibility of interoperability between different environments to give support to Web-based education. Based on this project and on the need to re-use the contents already present in the Web-based education servers, this work presents an object oriented framework that facilitates the description, localization and use of educational content available in servers compatible with the IMS project.

## Keywords

Architetures for educational technology system, computer-mediated communication, cooperative/collaborative learning, distance education and telelearnign, distributed learning environments.

## 1 Introduction

### 1.1 Motivation

The number of resources available on the Web continues to grow exponentially and with them the need to obtain more information about the resources that are available (IMS, 2000a). As a library uses various catalogues to find a book on a shelf, the Web also needs catalogues to find the resources available on the various servers. In the case of a library, the contents of these catalogues contain information about books, such as its author, title and publisher. In the case of the Web, the information contained in the catalogues is about the resources, which in the case of educational resources could be the title, description and format. This information about information is called meta-data (IMS 2000b; Metadata, 2000) or, specifically in the context we are dealing with, "information about educational content available in content servers."

With the description of the content using meta-data, the search for a content is facilitated if a search tool, which deals with this type of information, is used. With this tool, and given that the meta-data has the characteristics of the content available in the servers, the search for content becomes more efficient.

The ContentNet framework supports an educational content search tool based on meta-data. This tool, partially localized on the search server and partially on the content servers, establishes communication between the search and content servers, with the aim of allowing access to the meta-data and content available on these content servers.

## 1.2 Organization

This document is organized in six sections, with the first including an introduction. The second section describes the AulaNet environment. In Section 3, an introduction to the projects related to the presentation of contents is shown with emphasis on standard IMS.

The ContentNet framework is presented in Section 4. In Section 5, there are details about the instance of the framework. This section also deals with a study on other possible instances of the ContentNet framework. In Section 6, a comparison between a search done using the ContentNet and using a search-engine is shown. In the last section, Section 7, the conclusions reached on the work are presented as well as proposed work to be done.

## 2 The AulaNet Environment

The AulaNet (Lucena, Fuks *et al.*, 1998) is an educational environment for the creation, maintenance, assistance, and administration of Web based courses whose actors are: administrator, student and teacher (Lucena, Fuks *et al.*, 1999b). It offers a collection of communication, co-ordination and co-operation mechanisms to the teacher so that he can customize his course according to the objectives of the learning process. The communication mechanisms include tools for: sending messages to teachers, discussion group, interest groups, chat and contact between participants. The co-ordination mechanisms supply tools for: notices,  lesson plan, tasks, creation of evaluation and participation. The co-operation mechanism groups together: bibliography, webliography, documentation, teacher co-authorship, student co-authorship and a download facility (Lucena, Fuks *et al.*, 1999c).

While studying the resources available in these three types of mechanisms, it was possible to verify that the contents produced by the teacher refer to material, a lesson, documentation, a bibliography, a *webliography*, an evaluation, a question, a task and to a course as a whole. In more detail, AulaNet content is understood to be: material, or in other words, a file plus an explanatory text; a lesson, a collection of material plus an explanatory text; documentation, a collection of material plus an explanatory text; a bibliography, an explanatory text and a file; a *webliography*, an explanatory text and a link; an evaluation, a question and a task, which includes texts; and a course that includes the grouping of all the previously cited content .

In Table 1, one can see the three types of mechanisms that were mentioned and their resources. The resources that are considered as content are underlined. This highlighting does not include content material and questions as they are not AulaNet resources. They were only considered resources content due to the fact that material is the smallest representation of content composed of a file, material is present in any content that uses files, and the test question is the smallest representation of the evaluation content.

Because of the large number of information servers available and the dissemination of the AulaNet environment, which since July 2000 has seen more than 2,500 servers installed both inside and outside the country (Lucena, Fuks *et al.*, 1999a), the necessity arose to divide contents between the different servers. In other words, the need arose to use didactic resources that had already been published by different authors on different servers.

Given that the AulaNet environment is centered on courses, the level of granularity present was the course itself. In other words, it was no longer possible to access each existing item of content, but only the course as a whole. With the aim of making it possible to re-use all of the content present in AulaNet environments, independent of any hierarchy existing between them, it was necessary to alter the granularity to the smallest level that might contain semantics. In this way, the granularity is reduced to each AulaNet resource and to each content that makes up the course, permitting, for example, the re-use of a course, material or even a self-evaluation question. Once the granularity of the AulaNet content is defined, the next step is to catalogue the contents produced by the teachers.

The cataloguing is made necessary by the search for each desired content. The content researcher, a new role for the AulaNet teacher, uses the cataloguing attributes such as, for example, content title, description and format, to identify the desired content. In order to catalogue all of the content produced by teachers, it became necessary to create a new player, the cataloguer acting, on each AulaNet server.

The role of the cataloguer is to register the meta-data of existing content and update the database that stores this information as teachers create new content. The content meta-data registered by the cataloguer must follow the norms of the IMS standard since all content, AulaNet or not, must follow the same cataloguing standard.

## 3 Standards for the Representation of Contents

The concept of interoperability is not new. This concept originated in the last decade (Pathware, 2000), in the commercial aviation industry, with the “Aviation Industry CBT Committee” (AICC, 2000) being the first to contribute relevant specifications for the interoperability of systems.

Initiatives emerged with the intention of developing specifications establishing relationships with other initiatives that, as they come together, manage to resolve many problems and consequently develop specifications with greater speed and efficiency.

Among these initiatives are the AICC, the IMS, the IEEE LTSC (2000), the W3C (2000), the CedMA (2000), the ADL (2000), ARIADNE (2000) and the MMI Workshop (2000).

### 3.1.1. Description of the IMS Project

The "Instructional Management Systems Global Learning Consortium" (IMS) has been developing and promoting specifications to facilitate the distribution of educational activities such as how to find and use educational content and to trace the progress and report on the performance and the experiences of students in administrative and management systems.

The main problem with instruction management systems is that each one implements the management system functions in its own way. Because of this, for the developers of products and contents, development generally costs a lot for a variety of reasons (IMS, 2000c). With this in mind, the IMS project has been developing certain standards: "IMS Learning Resources Meta-data Specifications" (LRMS); "IMS Enterprise Specification" (ES); "IMS Content & Packaging Specification" (CPS); "IMS Question & Test Specification" (QTS) and "IMS Learner Profiles Specification" (LPS).

The IMS project joined other initiatives, such as ADL, which uses the specifications defined in the IMS project; to ARIADNE; to AICC; to Dublin Core and to W3C, which create low level specifications such as XML (2000), used by IMS.

### 3.1.2 Problems not yet resolved

The IMS project is quite new and therefor it is still in development. The first version of one of the specifications arrived, in August 1999, with the LRMS. Next, came the ES, the CPS and the QTS. The only specification that does not yet have a finished first version is the LPS.

The IMS project, as could be seen earlier, does not define the communication protocol used between systems that exchange content. More specifically, taking the search and exchange of content by a search tool as a base, the IMS project does not define the nature of the protocol used in a content search, nor the protocol used in the exchange of content between one system and another. This project concentrates on the specification of the description of the meta-data and on the description of the content itself to be exchanged, therefore not establishing a protocol that makes their interoperability viable.

### 3.1.3 What we shall use of the project

With a view to the interoperability of educational contents, the ContentNet framework is based on the LRMS. The framework uses the description of meta-data proposed in the IMS project to permit the search and exchange of educational content spread out among various content servers. From now, when the IMS standard is mentioned, this standard shall be understood as the description of meta-data.

# 4 ContentNet Framework

A framework is a collection of classes that represent the abstract design of a family of related problems. A framework is an architecture design that permits maximum re-use. It is represented by a collection of abstract and concrete groups encapsulated in such a way that sub-classes can be specialized for a given application (Mattsson, 1996).

A framework must be used as a starting point for the development of a collection of applications related to the same domain and should be seen as a collection of classes which possess re-usable functions for a specific domain of applications.

The kernel of the framework represents a similarity between the applications while the specific behavior is given by the hot spots (Fontoura, 1999). The kernel of the framework captures the general control of the application while the hot spots should be implemented to generate a instantiation of a certain domain of the application. The hot spots express aspects of the domain of the framework that could not be foreseen in its design.

The common elements in the process of the development of a framework are described in Figure 1:

The ContentNet framework was developed based on the process described above. Firstly, the domain of the application which would be constructed was analyzed, and, next, the first version of the framework was produced. In order to test the kernel and the designed hot spots, an instance of an application was generated. This application will be presented in the next section.

## 4.1 Description of the ContentNet framework

The ContentNet framework was designed to solve the problems related to finding, evaluating, providing access and manipulating information available in content servers. The aim of the ContentNet is to find content, based on its description, and transfer it from the server of origin to the destination server. To accomplish this, the ContentNet uses the concept of a network of servers that publish content and permit access to the content through a search server. The search server has access to the content servers using a protocol that is well known to them.

Figure 2 shows an example of the server network for the interoperability of content.

As previously seen, the objective of the ContentNet framework is the interoperability of content among all educational content servers that use the IMS standard. The process of interoperability of a content is complex and can be divided into three parts: finding the content, accessing it and registering it.

Finding quality content on the Web is a difficult task. In order to find content on the network, it is necessary to know how to give a good description of the desired content, and just as important, to know where to look for it. The content servers must be known and they must use the same standard for describing content. Once the content is found, it must be possible to access it. The server that has the content must supply an access mechanism, thus permitting that those searching for the content can acquire it. Together with the possibility of acquiring content, there must also be the possibility of registering it on another server, in this way achieving content interoperability.

## 4.2 The Architecture

In order to cover the interoperability of content, the ContentNet framework is composed of two large modules[1]: *Content Search* and *IServer*. The *Content Search* module, as its name suggests, acts as the search server while the *IServer* module acts as communication between the content server and search server. The *IServer*, *Intermediator Server,* sends the search server the information supplied

[1] Module: A planned unit designed to join or adjust itself to other analogue units to form a homogenous and functional whole

by the *Content Server* and sends the content server the information made available by the search server.

In order to generate an instance of the framework on any educational content server that uses the IMS standard, independent of the manner of data storage, it is necessary that when generating the instance, classes are implemented together with the *IServer* module that make the interface between the instance of the framework and the content server. The classes implemented on the hot spot of the framework must be capable of bringing information from the database of the *Content Server* to the instance of the *IServer* framework and to record information coming from the search in the *Content Server* database. This interface must provide information about the meta-data of a content and about the content itself. Figure 3 identifies the mentioned hot spots.

Due to the fact that it concerns a centralized solution, a *BackUp Server* is needed beacuse must assume the actions of *Content Search* every time it is not able to proceed.

The *Content Search* and *IServer* modules were divided into small parts conforming to the client-server architecture (Tanenbaum, 1995). The name of process—a program in execution—has been given to these parts. The server processes are designed to receive messages, while the client processes are designed to send messages. The servers are processes that are always active from the moment an instance of the module to which they belong is generated. The client processes are activated when their modules need to send a message.

The messages exchanged between the *Content Search* module and the *IServer* module are represented in Figure 4 and are listed in Table 2.

The hot spots, *CompareContent*, *GetContent* and *SetContent* are classes implemented in the instantiation of the *IServer* module according to the manner of storage of the meta-data and the content in the *Content Server* database. The hot spot *CompareContent* is designed to receive meta-data and supply a list of similar meta-data found in the database. The *GetContent* class must make available the content whose meta-data was received and the *SetContent* class must register the received content.

## 4.3 Diagram of classes and interaction

This section shows the diagram of the framework classes or, in other words, the diagram of the *Content Search* module and the diagram of the *IServer* module before its instantiation. Diagrams of the interaction of the client and server processes of the *IServer* module also will be shown. The diagrams of the interaction of the *Content Search* module processes have not been added as they are very similar to the diagrams of the interaction of the *IServer* module.

### 4.3.1 IServer Module

As has been seen in the earlier sections, the *IServer* module was divided into two parts, which have been given the names client process and server process. The description of the classes implemented in these two processes and their diagrams of interaction will follow, and lastly, the diagram of the *IServer* module class, marking the classes which include each process.

#### The Client Process

The aim of this process (Figure 5) is to send messages to the *Content Search* server. The *Client* class, the main class of this module, is responsible for the management of messages. In sending a message, the *Client* class communicates with the *ConnectionC* to establish a connection via socket with the server, and to create the communication channels through access to *ChannelC* class. When the connection is established and the communication channels created, the message is sent to one of the classes derived from the *Message* class, according to the type of message to be sent.

The ServerThread and the Channel classes presented in Figure 5 belong to the server process. Figure 6 shows the interaction diagram of the client process.

## Server Process

The server process (Figure 7) is responsible for attending requests made by the client module of *Content Search.*

The main *Server* class is responsible for establishing a connection with the *Content Search* client by means of the *ConnectionS* class, which creates a secondary process that remains responsible for attending requests. In this way, the main process comes back to wait for a new request while the secondary process, by means of a *ServerThread* class, is charged with the new request. The *ServerThread* class creates the communication channels with the client using the *Channel* class and checks the type of request that the *Content Search* client makes in the message that is sent. Depending on the message sent, different classes could be called as it is shown in Table 3. If the message *AskMD* is sent than the class *CompareContent* is called to compare the meta-data received to all meta-data stored on the database. If the message *AskCT* is sent than the class *GetContent* is caller to get the content that contemns a meta-data like to the meta-data that was received.  If the message *CT* is sent than the class *SetContent* is called to store the content that was received within the message.

In the case of the *CompareContent* or *GetContent* classes being called, the *ServerThread* then calls the *Client* process class to send the information returned by the classes. The diagram of the interaction of this process can be found in Figure 8.

In this way, the *CompareContent, GetContent* and *SetContent* classes are abstract classes. They do not possess implementation, therefore they must be implemented according to the instance of the framework. These classes, *IServer* module hot spots, are shown by circles in Figure 7.

With the aim of helping in the generation of an instance, the *incomplete, extensible* and *unique* stereotypes were used. The *incomplete* and *extensible* stereotypes were used according to the definition given by Fontoura (1999) and the *unique* stereotype was used according to UML(Booch, Rumbauch *et al.*, 1999).

### 4.3.2 Content Search Module

In the following paragraphs, the client and server processes, divisions made in the *Content Search* module as previously seen, are described, together with the classes used in each process. Lastly, there is a diagram of the *Content Search* module and the markings of the classes used in each process.

#### Client Process

The client of the *Content Search* module(Figure 9) is responsible for sending messages to the *IServer* server.

The client process of this module is very similar to the client process of the *IServer* module differing, though, in the messages sent. Table 4 shows the message sent by each class. *AskMD* message is sent by *MsgAskMD*, *AskCT* message is sent by *MsgAskCT* and *CT* message is sent by *MsgCT*.

Process Server

This process(Figure 10) is also very similar to the server process of the *IServer* module. The difference is in the treatment of the received messages. When the *"IP"* message is received together with the message, it is stored in the list of servers which form part of the ContentNet network. In other messages, *"AnsMD"* and *"AnsCT"*, this process simply stores the information received in places known to the cataloguer interface module described in Figure 4.

## 5 Instance of the ContentNet Framework

An instance of the ContentNet framework was generated with the aim of checking its viability, testing its design in respect of the definition of the kernel and hot spots, and testing its implementation. When generating an instance for the AulaNet, three AulaNet servers and one search server were used, as a centralized solution was adopted in the design of the framework.

The process of generating an instance of the framework foresees two stages. The first stage is the generation of the part of the framework that will constitute the search server, that is, the generation of the instance of the *Content Search* module. The second stage of the generating corresponds to the *IServer* module in the content servers that will form part of the ContentNet network. In the test carried out, 3 instances of the *IServers* were generated, one in each AulaNet server.

### 5.1 The Content Search Module

The generation of the instance of this module is simpler than that of the *IServer* module in relation to hot spots. As the *Content Search* module does not possess hot spots, their implementation is not necessary. However, the implementation of the search interface coupled to this module is necessary. This interface is implemented in order to supply the client process of *Content Search* with messages to be sent to the *IServers*. On generating the instance of the *Content Search* module, its server process prepares itself to receive messages sent by the *IServers*, only finishing when the search server is switched off. The client process only begins when a message needs to be sent to the *IServers*. Once the message has been sent, the client process finishes its execution.

The first message that the server process receives from the *IServer* module, or content server, is the message stating that it wishes to form part of the search network. Other exchanges of messages only occur after a *Content Search* user starts his search, stimulating the client process through the use of the interface.

## 5.2 The IServer Module

The instance of *IServer* module must be generated together with a content server. This module is designed to communicate between the search server and the content server. In this communication, the *IServer* module receives the messages from the search server and activates one of the implementations of the three hot spots, *CompareContent, GetContent* and *SetContent*, according to the message received. The server process starts its execution when the instance of the module is created and only finishes when the *IServer* module is stopped. The client process is activated by the server process when this wishes to send messages and is not activated by an action of the user.

Next, details about the implementation of the hot spots and details of the implementation used in AulaNet content servers will be shown. All of the three instances of *IServers* in the AulaNet servers had the same implementation in regard to the three hot spots.

### 5.2.1 CompareContent

The *CompareContent* class possesses the method *doCompare*, which receives a piece of meta-data and the type of content described in it as a parameter. The implementation of this method must compare the meta-data received with meta-data of the same type present in the content server database returning none or more meta-data equivalent to that one previously received.

The implementation of the hot spots must supply a class which possesses an inheritance relationship with the *CompareContent* class. During the generation, in accordance with the AulaNet server, the class implemented was *CompareAulaNetContent*. The *doCompare* method, implemented in accordance with the AulaNet, used an algorithm of comparison (Cormen, Leisersn *et al.*, 1999b) which compares chains of characters, strings.

### 5.2.2 GetContent

The objective of the *GetContent* class, the implementation of the *GetContent* hot spots, is to find the content in the database of the content server. This method receives the type of content desired and the meta-data of this content as a parameter and returns the content which has this description.

During the generation, in accordance with the AulaNet server, the class implemented was *GetAulaNetContent*. The *doGet* method checks what type of content it must look for and goes through the meta-data received searching for the attribute which identifies the content described by the meta-data. Having the identification and type of content it is possible to find it in the database.

### 5.2.3 SetContent

The objective of the *SetContent* class is to register content. The *doSet* method receives, as an entry parameter, the content to be registered and its type and must register the content received.

The implementation of this hot spot for the AulaNet servers creates the *SetAulaNetContent* class. In this case, the method *doSet* not only registers the content but also the explanatory text received in the message. Figure 11 shows the diagram of classes of the server process of this module with the implementations of the hot spots.

The generation of the instance of the *IServer* module for AulaNet content creates another requirement in addition to the implementation of hot spots. There is the need of a tool for register meta-data about the content present in the AulaNet database.

### 5.2.4 Tool for registering meta-data

The AulaNet environment used in testing the ContentNet framework had to be adapted. The ContentNet framework is based on the IMS standard, although the AulaNet does not follow this standard. The standard presupposes that all content registered possesses related meta-data, an example is shown in Annex 1. To adapt the AulaNet environment to the standard, the meta-data registration tool was created. This tool, which does not form part of the ContentNet framework, permits the cataloguer to register meta-data for all the content present in the database. The tool acts in two stages as shown in Figure 12.

During the process of registering the meta-data, the system supplies the cataloguer with a form containing the meta-data attributes. In the first stage, the cataloguer fills in the attributes and sends the form to the system. The meta-data attributes which require knowledge of the content, are registered by the cataloguer. In the second stage, the system adds to the meta-data, filled in by the cataloguer, the content identification attribute, for example the attribute `size` of `technical` in Annex 1. This attribute is catalogued by the system, as it is a piece of data that only the system knows.

## 5.3 Message flow diagram

The instance of *Content Search* module is generated before the first *IServer* module. The server process of the *Content Search* module starts its execution before the *IServer* module server process. Figure 13, shows the diagram of the flow of messages exchanged between the *Content Search* and *IServer* modules.

## 5.4 Other instances

When generating the AulaNet instance of the ContentNet framework, its hot spots were implemented in accordance with the characteristics of the AulaNet environment. The implementation of these hot spots could undergo a lot of alteration if their instance were generated in other environments. The change in the implementation of the hot spots would be as great as the differences between the AulaNet environment and the environment where the instance would be generated.

Details about factors which can influence the implementation of the hot spots follow, as described in Table 5.

The storing of information (contents) and meta-data (descriptions of contents) are factors which influence the implementation of the hot-spots. The manner of storing these items depends on the implementation of the environment which can be as different as possible from the manner of storing items in the AulaNet environment. Consequently, the search for and acquisition of the information and meta-data depends on the implementation of the environment.

As an example, in the AulaNet environment, meta-data is stored in files. The database only supplies the nature and location of the meta-data file of the content. This means that to access an attribute of the meta-data, it is necessary to go to the database, collect the data (the nature and location of the meta-data) and, next, go through the meta-data file until the desired attribute is found.

Another option for storing meta-data is not to use files and register to register it in the database. The attributes in the table of the database correspond to the attributes of the meta-data. Each entry, or line, of the table corresponds to a piece of meta-data. Thus, activating an attribute means finding the meta-data desired and, for that line, "looking" at the column of the table which corresponds to the attribute.

Next there is an analysis of the other possible implementations of the hot-spots *CompareContent*, *GetContent* and *SetContent* for environments which have little similarity to the AulaNet.

### 5.4.1 CompareContent

As well as the manner of storing meta-data, the method of comparison used in the implementation of this hot-spot can be different from that used for the AulaNet. In the case of the AulaNet, meta-data of the contents which form part of the search are selected. For these, the method used compares the string of meta-data received with the string of meta-data for the same attribute in the database, checking if they are similar. Another method could check if the string are equal. In this way, only meta-data which is exactly alike would be considered. Yet another option would be to use any of the other methods found in (Cormen, Leisersn *et al.*, 1999a), for example.

### 5.4.2 GetContent

The manner of storing the meta-data influences this hot-spot as much as the manner of storing the content. Another factor which influences its implementation is how the content is identified by its meta-data. In the case of the AulaNet, the content is identified by an attribute of the meta-data. In a new implementation of this method, in order to identify a content, it may be necessary to go through all the meta-data received comparing it with all of the meta-data present in the database (or in a file according to the manner of storage).

Another difference is in the composition of the content. For the AulaNet, content may be composed of more than one file, one of them being created dynamically, through consulting the database. For another implementation, content can be composed of only one file, without the need for dynamic creation.

### 5.4.3 SetContent

In this hot-spot, as its task is extremely simple, there is no other point which may influence its implementation apart from the manner of storing content. In the case of the AulaNet, a field identifying if the content was acquired through a search or if it was created in the AulaNet is associated with the content stored. The AulaNet does not make content acquired available for a search. Only content created in the AulaNet itself is available for a search. In the case of another implementation it may be decided that even content acquired through a search is made available for a search.

## 6 Comparison between ContentNet and a search-engine

As shown in Annex 1, the meta-data structure developed by the IMS standard and used on ContentNet is detailed and complex. The standard characterizes a content by innumerable attributes organized in nine groups: metametadata, general, lifecycle, technical, education, rights, relation, annotation and classification. Each group is responsible for describe a set of characteristics of the content. For example, technical group describes technical characteristics of the content like format, size and minimum installation requirements and education group describes educational and pedagogical characteristics like educational level of the content user and degree of difficulty in using the content.

When an user is doing a search using the ContentNet, the user is allowed to use all attributed specified on the standard to describe the content that he/she wants. As doing so, the user could use attributes like title and author, specified on general group, and also format and educational level. For example, the user could look for a *HTML* content titled *Physic I* by *Jorge Fontes* that should be used for a *undergraduate student*.

On the other hand, when using a search-engine is impossible to do a search so specified as using the ContentNet.  When using a search-engine one can’t describe the format of the content neither the educational level of the user content. Therefore localize and obtain a specific content using the ContentNet is easer than using a search-engine that is not based on a meta-data information.

## 7 Conclusion and futures work

Using the IMS project as a base, it is possible to construct environments that deal with the meta-data available in content servers. Through the standardized meta-data of the contents, it is possible to find content desired content more efficiently. Having the information that the desired content exists, the transfer of this content between two systems that communicate with each other becomes viable.

The ContentNet framework provides support to the creation of a tool which permits the exchange of educational content between content servers that are in accordance with the IMS standard. The proposed framework is aimed at facilitating and making it possible to search for and transfer of these contents.

By implementing the hot spots of the framework, it is possible to adapt it in relation to the form of storing the contents that have been exchanged, to the method used in the comparison of meta-data and to the identification of a content, given its meta-data.

Due to the compatibility of the enhanced AulaNet servers with other servers that use the same concept of meta-data proposed in the IMS standard, it is possible to create a community of knowledge centered on educational content.

The development of the communication protocol between the content servers and a search server contributed to the development of the AulaNet environment as to the evolution of the IMS project, of which we are participants. It added to the AulaNet a tool that permits teachers to search for and find content. To the IMS project it added a tool that establishes the protocol for the exchange of content.

The prototype version was developed autonomously and is now in a phase of integration with the AulaNet in version 2.0.

## 6.2 Future Projects

- Owing to the possibility of up-dating the version 1.0 of the meta-data proposed by the IMS project, it is necessary to create a tool or make an adaptation to the framework designed to allow the creation of new elements for meta-data and the removal of obsolete elements. To support this modification - the creation and exclusion of elements of meta-data - the IMS standard created the TIERS structure repository. Using TIERS as a base, the creation of the tool and the adaptation of the framework are made easier.

- The IMS standard foresees the sale of content exchanged between content servers, which does not happen in the case of ContentNet. The ContentNet does not deal with the sale of content, but only with the exchange of content between one server and another. Thus, it would be an improvement to the ContentNet framework to develop an interface with a tool that would sell the content between the server that possesses the content and the sever that desires it before its effective exchange.

- Currently the user of the *Content Search* framework has no alternative between acquiring content searched for and pointing to the content found. If these options were available, the user who wished to accompany the updating of a content could opt to use a link to the content, instead of acquiring it and storing it. The alteration of the framework to include this option would be of great value.

- With the aim of validating the framework design in relation to its hot spots and its kernel, an instance of it must be generated for other applications, with characteristics very different than the characteristics of the AulaNet environment.

## Acknowledge

This work has been supported by CNPq/Brazil under grant No. 140646/2000-0 for Viviane T. da Silva, No. 300031/92-0 for Carlos J. P. de Lucena and No. 524557/96-9 for Hugo Fuks;. MSc.Viviane, Phd. Lucena and Phd. Fuks are also supported by the PRONEX Project under grant 7697102900.

## Reference:

ADL (2000) Advanced Distributed Learning. [online].<http://www.adlnet.org>[Consultation: November 13th of 2000]

AICC (2000) Aviation Industry CBT Committees. [online]. November 06th of 2000. <http://www.aicc.org>[Consultation: November 13th of 2000]

ARIADNE (2000) Alliance of Remote Instructional Authoring & Distribution Networks or Europe [online]<http://ariadne.unil.ch/project/main.content.html>[Consultation: November 10th of 2000]

CedMA (2000) Computer Education Management Association.[online]. <http://www.cedma.org>[Consultation: November 09th of 2000]

Classnet. (2000) Iowa State University Computation Center.[online]. <http://classnet.cc.iastate.edu>.[Consultation: October 10th of 2000]

Cormen, T. H.; Leiserson C. E. & Rivest R. L.(1999a) *Introduction to Algorithms* London: McGraw-Hill.

Cormen, T. H.; Leiserson C. E. & Rivest R. L.(1999b) *Introduction to Algorithms* London: McGraw-Hill, 301-328

Fontoura, M. F. M. C.(1999), A systematic approach to framework development. Doctor Thesis, Department of Computer Science, Pontifical Catholic University of Rio de Janeiro.

IEEE LTSC (2000) IEEE Learning Technology Standers Committee.[online]. September 11th of 2000. <http://ltsc.ieee.org> [Consultation: September 20th of 2000]

IMS (2000a) IMS Metadata. [online]. IMS Web Team, October 16th of 2000. <http://www.imsproject.org/metadata> [Consultation: October 30th of 2000]

IMS (2000b) IMS Global Learning Consortium, Inc. [online]. IMS Web Team, November 07th of 2000. <http://www.imsproject.org> [Consultation: November 10th of 2000]

IMS (2000c) IMS Web Team [online] October 4th of 2000 <http://www.imsproject.org/faqs.html> [Consultation: November 04th of 2000]

Lucena,C.J.P.; Fuks,H.; Milidiu,R.; Macedo,L.; Santos,N. & Laufer,C. et al. (1998). AulaNet—An Environment for the Development and Maintenance of Courses on the Web. In:ICEE'98—International Conference On Engineering Education, Rio de Janeiro.

Lucena, C.J.P.; Fuks, H.; Milidiu,R.; Laufer,C.; Blois,M. & Choren,R. et al.(1999a). AulaNet technologies: future trends. In:ICECE'99- International Conference on Engineering and Computer Education, Rio de Janeiro.

Lucena,C.J.P.; Fuks,H.; Milidiu,R.; Laufer,C.; Blois,M. & Choren,R. et al.(1999b) AulaNet: helping teachers to do their homework. In:Annals of XXVI Semish–Integrated Seminar of Hardware and Software, Brazilian Society of Computer(SBC), Rio de Janeiro.

Lucena, C. J. P.; Fuks, H.; Milidiu, R.; Laufer, C.; Blois, M. & Choren, R. et al.(1999c). AulaNet and new information technologies applied to the Web based education. V International Congress of Distance Education, Brazilian Association of Education(abed), Rio de Janeiro.

Mattsson, M.(1996). Object-Oriented Frameworks: A Survey of Methodological Issues. ,M.Sc. Dissertation, Department of Computer Science and Business Administration,University College of Karlskrona/Ronneby.

Metadata (2000), Metadata at W3C [online]. Ralph Swick, October 07th of 2000. <http://www.w3.org/metadata> [ Consultation: October 12th of 2000]

MMI (2000) Metadata for Multimedia Information [online] Luc Van den Berghe,August 20th of 1998 <http://www.cenorm.be/isss/workshop/mmi> [Consultation: March 06th of 2000]

Pathware (2000) Pathware 4 Attain.[online].Macromedia, Inc. <http://www.macromedia.com/software/pathware/productinfo/whitepapers/open_standards.html> [ Consultation: January 19th of 2000]

Tanenbaum, A. S.(1995) *Distributed Operating Systems*. New Jersey: Prentice Hall (pp.: 50-68)

Booch,G.; Rumbauch,J. & Jacobson,I..(1999) *Unified Modeling Language*. Massachusetts: Addison Wesley Longman, Inc. (1-482)

Virtual-U (2000) Simon Fraser University.[online].<http://vitual-u.cs.sfu.ca/ vuweb>. [Consultation: November 13th of 2000]

W3C (2000) World Wide Web Consortium.[online]. November 11th of 2000. <http://www.w3.org> [Consultation: November 13th of 2000]

Web-Course-in-a-Box (2000) Virginia Commonwealth Center.[online]. <http://views.vcu.edu/wcb>. [Consultation:November 10th of 2000]

WebCT (2000) University of British Columbia.[online].<http://homebrew.cs. ubc.ca/webct>. [Consultation: November 10th of 1999]

XML (2000) eXtensible Markup Language.[online].Dan Connolly, October 10th of 2000. <http://www.w3.org/XML/> [ Consultation: October 20th of 2000]

## Figure Caption

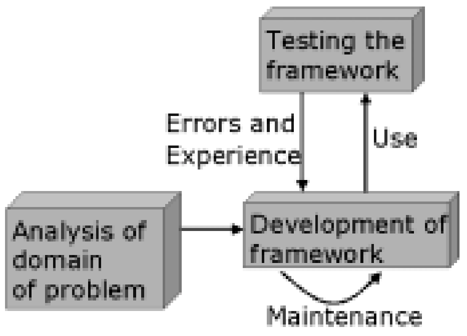

Figure 1 - The process of development of a framework (Mattson, 1996)

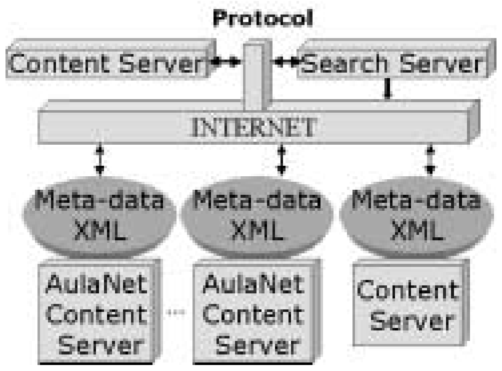

Figure 2 - Interoperability of content

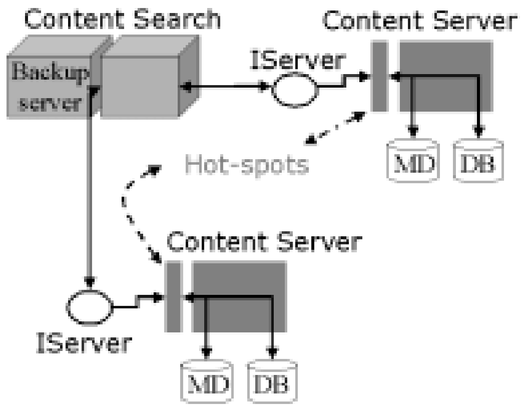

Figure 3 - Hot spots. MD- meta-data, DB- content base

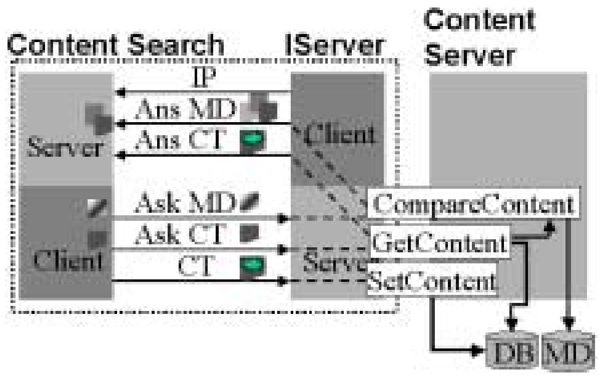

Figure 4 - Hot spot classes: CompareContent, GetContent, SetContent

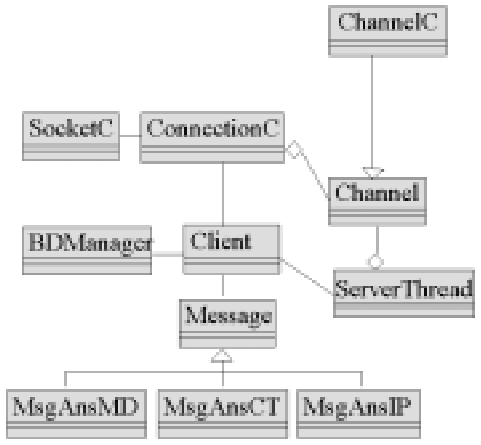

Figure 5 – IServer Module – cliente process

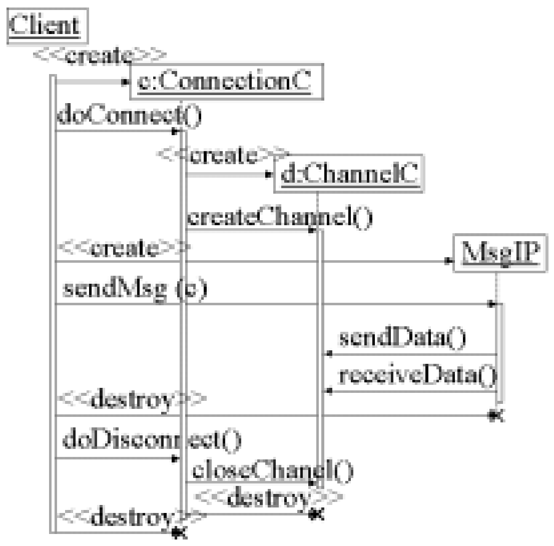

Figure 6 - Diagram of Interaction (client process)

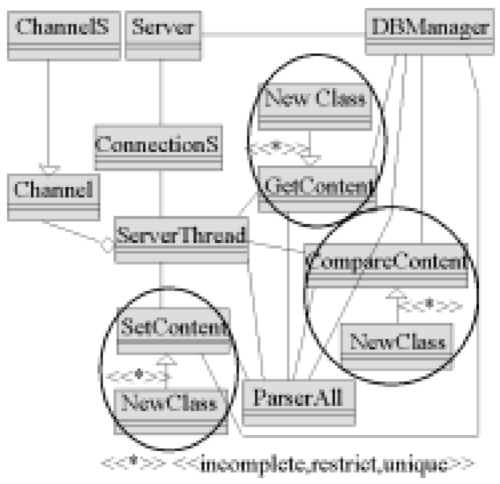

Figure 7 - IServer Module – server process

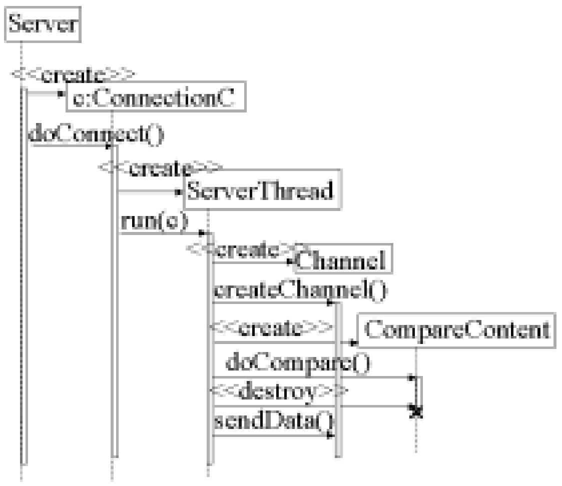

Figure 8 - Diagram of Interaction (server process)

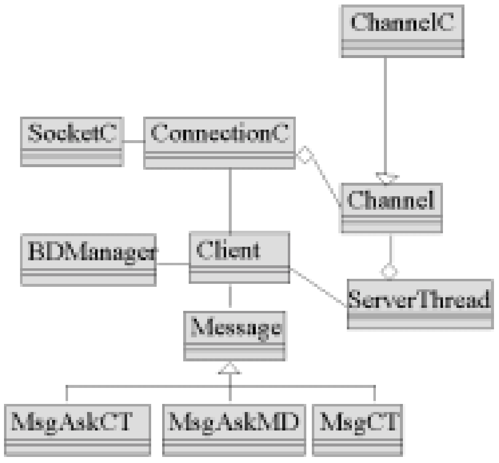

Figure 9 - Content Search Module – client process

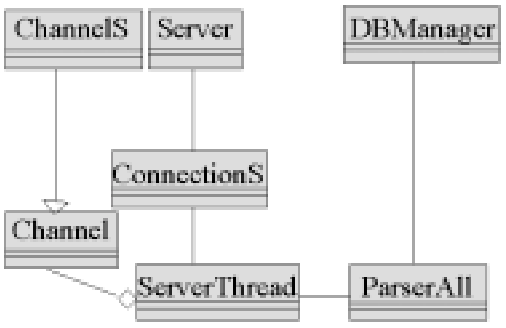

Figure 10 - Content Search Module – server process

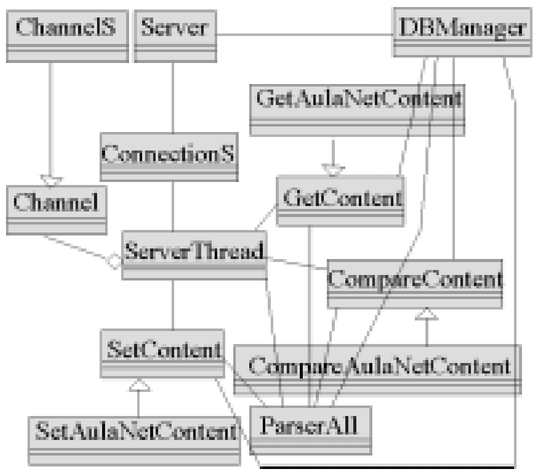

Figure 11 - IServer module – server process

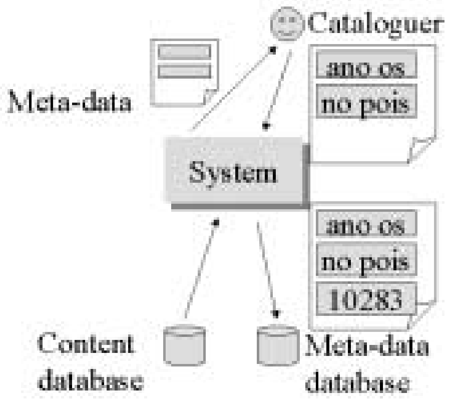

Figure 12 - Cataloguer Tool

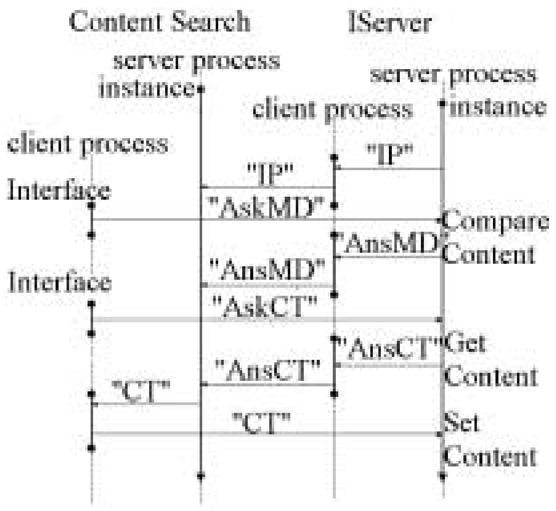

Figure 13 - Diagram of flow of messages

## Table Caption

| Mechanisms | Resources |
|---|---|
| *Co-ordination* | Tasks, Notices, Evaluation, Lesson plans, Accompanying participation |
| *Co-operation* | Bibliography, Webliography, Documentation, Teacher co-authorship, Student co-autorship, Download |
| “*CT*”, | Teacher’s email, Paticipants’ email, Internet Group, Discussion Group, Chat |

Table 1 - Mechanisms and their resources

| Message | Explanation |
|---|---|
| “IP” | Sent by *IServer* when its instance is generated and the *Content Server* wishes to be part of the search network of *Content Search* |
| “AskMD” | Question sent by *Content Search* to all *IServers* asking for similar meta-data to that the user desires. The *IServer* receives the meta-data and contacts the implementation of the hot spot *CompareContent* which supplies the similar meta-data found in the *Content Server* database |
| “AnsMD” | Message sent by the *IServer* to the *Content Search* as a reply to the “AskMD” question, supplying the list of similar meta-data |

| | |
|---|---|
| “AskCT” | Request for a determined content described in the meta-data which is being sent together with the message. The *IServer* that receives the message sent by *Content Search* contacts the implementation of the hot spot *GetContent* which makes the content possessing such meta-data |
| “AnsCT” | Message sent by the *Server*, supplying the content to *Content Search* |
| “CT” | Message which carries the *Content Search* content to the *IServer* that desires the content. The *IServer* contacts the implementation of the hot spot *SetContent* sending the content that is registered in the *Content Server* database. |

Table 2 – Message exchanged between the Content Search and the IServer module

| Message | Class called |
|---|---|
| *“AskMD”* | *CompareContent* |
| *“AskCT”,* | *GetContent* |
| *“CT”,* | *SetContent* |

Table 3 – Class that could be called

| Message sent | Class |
|---|---|
| *“AskMD”* | *MsgAskMD* |

| | |
|---|---|
| *“AskCT”* | *MsgAskCT* |
| *“CT”* | *MsgCT* |

Table 4 – Messages sent by the client process

| Hot spots | Factors that can influence implementation |
|---|---|
| *CompareContent* | The manner of storing the meta-data |
| | Method of comparison used in meta-data |
| *GetContent* | The manner of storing the meta-data |
| | The manner of storing the content |
| | Method of identification of the content by means of its meta-data |
| *SetContent* | The manner of storing the meta-data |

Table 5 - Factors that can influence implementation

| Group | Set of characteristics |
|---|---|
| Metametadata | |
| General | General information about the content like title, description and keyword. |
| Lifecycle | |
| Technical | |

| |
|---|
| Education |
| Rights |
| Relation |
| Annotation |
| Classification |

Table 6 - Groups and theirs set of characteristics

## Annex 1

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

<RECORD>
  <METAMETADATA>
    <CATALOGENTRY>
      <CATALOGUE>Content-Test</CATALOGUE>
      <ENTRY>
        <LANGSTRING>Content.1001.2</LANGSTRING>
      </ENTRY>
    </CATALOGENTRY>
    <CONTRIBUTE>
      <ROLE>
        <LANGSTRING lang="en">Author</LANGSTRING>
      </ROLE>
      <CENTITY>
        <!-- Simple vCard example -->
        <VCARD>
BEGIN:VCARD
N:Silva;Viviane.
FN:Carlos Lucena, Ph.D.
ORG:LES;
ADR:Brazil
TEL:+55
EMAIL;INTERNET:viviane@les.inf.puc-rio.br
LABEL;QUOTED-PRINTABLE:LES
END:VCARD
             </VCARD>
      </CENTITY>
      <DATE>
        <DATETIME>2000-02-01</DATETIME>
      </DATE>
    </CONTRIBUTE>
    <METADATASCHEME>IEEELOM:1.0</METADATASCHEME>
    <LANGUAGE>en-US</LANGUAGE>
    <!-- English as default metadata language.  -->
  </METAMETADATA>
  <GENERAL>
    <TITLE>
      <LANGSTRING lang="en-US">ASP Course</LANGSTRING>
    </TITLE>
    <CATALOGENTRY>
      <CATALOGUE>ISBN</CATALOGUE>
      <ENTRY>
        <LANGSTRING>0-222-33333-4</LANGSTRING>
      </ENTRY>
    </CATALOGENTRY>
    <LANGUAGE>en-US</LANGUAGE>
    <DESCRIPTION>
      <!--English description-->
      <LANGSTRING lang="en-US">Course for web programmers</LANGSTRING>
    </DESCRIPTION>
    <KEYWORDS>
      <!--English Keywords, unordered list-->
      <LANGSTRING lang="en">ASP</LANGSTRING>
      <LANGSTRING lang="en">course</LANGSTRING>
      <LANGSTRING lang="en">Web</LANGSTRING>
      <LANGSTRING lang="en">programmers</LANGSTRING>
    </KEYWORDS>
    <COVERAGE>
      <LANGSTRING/>
    </COVERAGE>
    <STRUCTURE>
      <LANGSTRING>Hierarchical</LANGSTRING>
    </STRUCTURE>
    <AGGREGATIONLEVEL>2</AGGREGATIONLEVEL>
  </GENERAL>
  <LIFECYCLE>
    <VERSION>
      <LANGSTRING>4.5</LANGSTRING>
    </VERSION>
    <STATUS>
      <LANGSTRING lang="en">Final</LANGSTRING>
    </STATUS>
    <CONTRIBUTE>
      <ROLE>
        <LANGSTRING lang="en">Technical Implementer</LANGSTRING>
      </ROLE>
      <CENTITY>
        <VCARD>
BEGIN:vCard
FN:Melissa Drugos
N:Drugos
END:vCard
             </VCARD>
      </CENTITY>
      <DATE>
        <DATETIME>1999</DATETIME>
      </DATE>
    </CONTRIBUTE>
  </LIFECYCLE>
  <TECHNICAL>
    <FORMAT>PPT</FORMAT>
    <SIZE>1032353</SIZE>
    <LOCATION type="URI">http://www.les.inf.puc-rio.br/~asp</LOCATION>
    <REQUIREMENTS>
      <TYPE>
        <LANGSTRING lang="en">Presentation</LANGSTRING>
      </TYPE>
      <NAME>
      <LANGSTRING lang="en">Power Point</LANGSTRING>
    </NAME>
    <MINIMUMVERSION>5.0</MINIMUMVERSION>
    <MAXIMUMVERSION/>
  </REQUIREMENTS>
  <INSTALLATIONREMARKS>
    <LANGSTRING lang="en">Load from diskette</LANGSTRING>
  </INSTALLATIONREMARKS>
  <OTHERPLATFORMREQUIREMENTS>
    <LANGSTRING lang="en">Run in browser.</LANGSTRING>
  </OTHERPLATFORMREQUIREMENTS>
  <DURATION/>
</TECHNICAL>
<EDUCATIONAL>
  <INTERACTIVITYTYPE>
    <LANGSTRING>Passive</LANGSTRING>
  </INTERACTIVITYTYPE>
  <LEARNINGRESOURCETYPE>
    <LANGSTRING lang="en">Presentation</LANGSTRING>
  </LEARNINGRESOURCETYPE>
  <INTERACTIVITYLEVEL>1</INTERACTIVITYLEVEL>
  <SEMANTICDENSITY>2</SEMANTICDENSITY>
  <INTENDEDENDUSERROLE>
    <LANGSTRING lang="en">Learner</LANGSTRING>
  </INTENDEDENDUSERROLE>
  <LEARNINGCONTEXT>
    <LANGSTRING lang="en">Tecnical</LANGSTRING>
  </LEARNINGCONTEXT>
  <TYPICALAGERANGE>
    <LANGSTRING>18-99</LANGSTRING>
  </TYPICALAGERANGE>
  <DIFFICULTY>5</DIFFICULTY>
  <TYPICALLEARNINGTIME/>
  <DESCRIPTION>
    <LANGSTRING lang="en">Follow the sequency of the presentations.</LANGSTRING>
  </DESCRIPTION>
  <LANGUAGE>en_US</LANGUAGE>
</EDUCATIONAL>
<RIGHTS>
  <COST>
    <LANGSTRING lang="en">yes</LANGSTRING>
  </COST>
  <COPYRIGHTOROTHERRESTRICTIONS>
    <LANGSTRING>yes</LANGSTRING>
  </COPYRIGHTOROTHERRESTRICTIONS>
  <DESCRIPTION>
    <LANGSTRING lang="en">Copyright 1999 Viviane Silva</LANGSTRING>
    <LANGSTRING lang="en">Contact publisher to purchase</LANGSTRING>
  </DESCRIPTION>
</RIGHTS>
<RELATION>
  <KIND>
    <LANGSTRING lang="en">IsReferencedBy</LANGSTRING>
  </KIND>
  <RESOURCE>
    <DESCRIPTION>
      <LANGSTRING lang="en">Companion book titled "ASP".</LANGSTRING>
    </DESCRIPTION>
  </RESOURCE>
</RELATION>
<ANNOTATION>
  <CENTITY/>
  <DATE/>
```

```
    <DESCRIPTION>
      <LANGSTRING lang="en-US">A useful presentation for be used on courses of programming
languages for web.</LANGSTRING>
    </DESCRIPTION>
  </ANNOTATION>
  <CLASSIFICATION>
    <PURPOSE>
      <LANGSTRING lang="en">Discipline</LANGSTRING>
    </PURPOSE>
    <TAXONPATH>
      <SOURCE/>
      <TAXON>
        <ID>300</ID>
        <ENTRY>
          <LANGSTRING lang="en">Social Sciences</LANGSTRING>
        </ENTRY>
      </TAXON>
    </TAXONPATH>
    <DESCRIPTION/>
    <KEYWORDS/>
  </CLASSIFICATION>
</RECORD>
```