

Distributed Virtual LABoratory

DiViLab

Active Document Manual

Authors	Verdejo, M.F.; Barros, B.; Read, T.; Mayorga, J.I, Vélez, J., Calero, M.Y. (UNED); Viéville, C. (USTL); Pinkwart, N., Hoppe, U., Schmidt, P. (UDUI)
Emitting Partner	P5-UNED
Responsible	
Validated by	
Sent for information only	
Sent for action	
Deliverable n°	7.6
Workpackage n°	WP7
Cost	
Document is public ?	<input type="checkbox"/>
For EC ?	<input type="checkbox"/>

Contents

1	Introduction	5
2	Installation of an AD Server	6
2.1	System requirements	7
2.2	Installing the program.....	7
2.2.1	Copy of Installation files	7
2.2.2	Copy of libraries	8
2.2.3	Running the installation program	9
2.2.4	Updating after the changes	17
2.2.5	Possible errors produced during installation	18
2.2.5.1	Connection errors with the ADServerSetup	18
2.2.5.2	Format errors in the values entered by the user.....	18
2.2.5.3	Context errors provoked by previous installations	18
2.2.5.4	Errors with the DB.....	19
2.2.6	Configuration file	19
2.2.6.1	Elements for configuring the AD environment	19
2.2.7	Un-installation of the ADServer.....	30
2.3	Installing the Task Manager	31
3	Installation of an AD client.....	33
3.1	System requirements	33
3.2	Installing the program.....	33
3.2.1	Navigator	33
3.2.2	Java plugin.....	34
3.2.3	SVG plugin	34
3.3	Creation of a new AD	34
3.3.1	Errors when a new AD is created	44
3.4	Description of the AD XML files.....	44
4	Using and Configuring the AD system.....	46
4.1	Working in the system as a learning scenario designer.....	47
4.1.1	The XML files which are used to define a scenario with the AD	52
4.1.1.1	Tool types	53
4.1.2	The DescriptionAD	53
4.1.2.1	AD elements	53
4.1.2.2	Editing the AD-definition XML file.....	64
4.1.2.3	Definition of prerequisites	75
4.1.2.4	Defining internal tools.....	82

4.1.2.5	Defining the correction tool	89
4.1.3	Definition of the resources	93
4.1.3.1	Elements for defining a resource	93
4.1.3.2	Editing a resource XML file	96
4.1.4	Definition of the community	104
4.1.4.1	Elements for defining a community	105
4.1.4.2	Editing a community XML file	109
4.1.5	The results	119
4.1.5.1	What is the results file?	119
4.2	Working on an AD scenario as a student	120
4.2.1	Login the system	121
4.2.2	The interface	123
4.2.3	How to interact with the environment	124
4.2.3.1	Use of internal tools within the tasks	127
4.2.3.2	Navigation menu	128
4.2.3.3	Tool Bar	132
4.2.3.4	Prerequisites revisited	134
4.2.4	Errors that can be produced when using the system in the role of student 134	
4.3	Working on an AD scenario as a teacher	137
4.3.1	Login the system	137
4.3.2	The Monitor	139
4.3.3	The AD Corrector	145
4.3.4	How to interact with the correction tool	146
4.3.4.1	Marking	146
4.3.4.2	Viewing	150
4.3.4.3	Details of the correction tool	152
4.3.5	Errors that can happen when using the system with the role of teacher	152
4.4	Eliminating an AD	153
5	Application	153

1 Introduction

This document presents the ActiveDocument (AD) System from two different viewpoints, firstly, that of the installation, the system requirements, the actual installation process, and the possible errors that can occur. Secondly, that of a user, which in itself can be divided into three types: a scenario designer, a student, and a teacher. Each of which will be detailed together with the option and functions available. For a more general view of the AD System, its architecture and the way scenarios are built, the reader should consult the DiViLab deliverable D7.5, and for more specific information about the chemistry scenario mentioned here, the reader should consult the UNED technical report TR1.

The Active Document enables collaborative learning activities to be carried out, using a variety of tools and resources, in a distributed framework. It has been implemented with Java and XML technologies and has an underlying client/server architecture. The server runs on a SUN server with Tomcat, msQL and Java. Any client can connect to the AD services using a standard web browser. Emphasis is placed on learning experimental sciences where there is a pressing need for students to improve their learning processes with a better understanding of theory and practice throughout the academic year, especially in the context of distance learning. A way forward is to engage the students in a variety of activities, including the performance of experiments either in real or virtual settings, supported by a distributed collaborative computer environment. The premise here is to offer a persistent, structured, dynamic, active and personal work space to sustain their constructs in a long term learning process.

In the context of the DiViLab project the ActiveDocument (which has been developed during the project) is being extended to make use of related components developed by other groups in the project, namely the Task Manager (developed by USTL) and the Monitor (developed by UDUI). The objective of the former is to reflect the state of the work by building views. This state is built according different points of view and according, at the same time, the level of detail required by the client which requires access to different views.

The relation between the Active Document and the Task Manager and Monitor can be seen in figure 1. The former receives both the declaration of the experiment and its participants before an experiment is started (and then as the experiment is undertaken, the ongoing student progress) and the latter permits the teacher or tutor to interact with this information and see different views of it. It should be noted, that the Task Manager is transparent to the user, is does not as such present a direct user interface when used from the Active Document. The students experience its action in terms of permissions to advance from one task to the next within the experiment and the teachers in terms of the views provided to the Monitor.

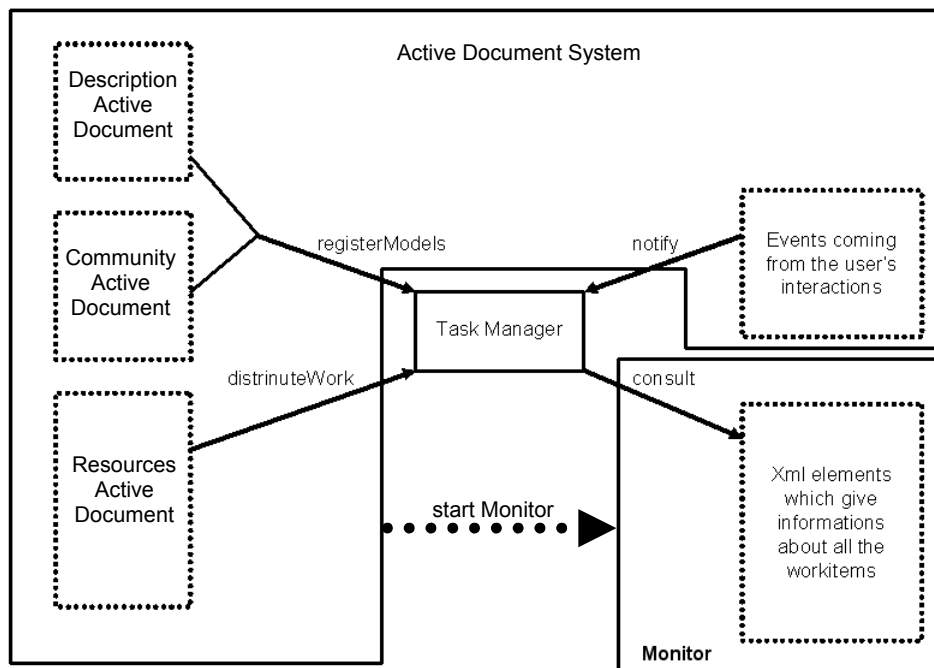


Figure 1. The DiViLab prototype

Now, the task manager is ready to reflect the users' interactions with the all the resources within a course. Each time the user makes a significant interaction which can change the state of a workitem, the ActiveDocument sends an event to the Task Manager to maintain the state of the scenario.

The Monitor is being developed by UDUI that gives tutors a fast and precise overview enables them to get informed of students with minor or greater problems solving the tasks. It is important that teachers know about that so they are able to react in a fashion they like. In the same way students with great performance can be easily found and perhaps be contacted to help others with needs.

In chapter 2, the installation of the AD server is detailed together with system requirements. In chapter 3, the installation of the AD client is detailed, reflecting not only the AD tools but also the plugins and associated applications that are required. Subsequently, in chapter 4, the way in which the AD system can be used is presented from three different views, that of designer, student, and teacher.

2 Installation of an AD Server

Throughout this section, the hardware and software requirements necessary to install the server program for the Active Document will be explained.

2.1 System requirements

Hardware:

- 512 MB RAM

Operating Systems:

- Windows
- Linux
- Solaris

Software:

- Programs:
MySQL 3.23.49
j2sdk1.4
Tomcat 4.1.18
- Required library programs:
Xerces 2.0.1 → xercesImpl.jar and xmlParserAPIs.jar
Xalan 2.4.0 → xalan.jar

2.2 Installing the program

The installation program is a web application and we will be using Tomcat as a web server to run it.

2.2.1 Copy of Installation files

First of all, the installation program must be copied on the machine where the Active Document Server is being installed.

Since the installation program works as a web application and Tomcat will be used as a web server to run it, the installation program must be unzipped in a specific directory within the structure of the Tomcat folders, specifically at the directory “webapps”.

The installation process will generate a folder called “ADServerSetup” or a new context within Tomcat called “ADServerSetup”.

Next we describe the process to be followed depending on the operating system we have installed.

In Windows:

Run ADServerSetup.exe, and a window like in figure 2 will show up.

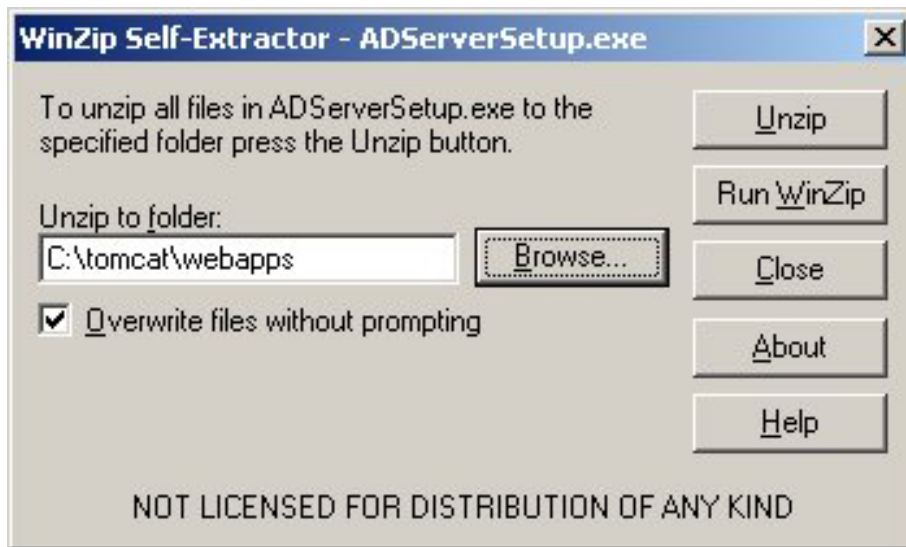


Figure 2. Decompressing the installation file

1. Click on Browse to indicate where to unzip the files within the folder “webapps” from the tree of directories from Tomcat whose location will depend on where it is installed in our machine. (In our case it is in c:/tomcat/webapps).
2. Click on unzip to proceed to unzipping.

In Linux and Solaris:

To unzip the installation package ADServerSetup.tar.gz within the directory webapps of Tomcat:

1. Copy the file ADServerSetup.zip within the directory webapps of Tomcat.
2. Run: `unzip ADServerSetup.zip`
3. Ensure that the directory ADServerSetup (and everything under it) is owned by the user that tomcat will run as (e. g. the tomcat user).

2.2.2 Copy of libraries

In order for the installation program to work, it is very important that some determined jar files which are included in certain libraries be included in the Tomcat directory.

These files are:

- Xerces 2.0.1 → xercesImpl.jar and xmlParserAPIs.jar
- Xalan 2.4.0 → xalan.jar

If they aren't there, we will lose a copy of said files inside the directory common/lib of Tomcat.

In the case of having files pertaining to prior versions of these same libraries, we will proceed to eliminate and copy the indicated files in order to avoid problems of compatibility.

2.2.3 Running the installation program

The process of installation of the Active Document Server contains a series of steps; we need some determined steps to be active, to connect to the installation program thorough the navigator, to enter the configuration parameters necessary for the ADServer, and finally to carry out the copying of the files that compose the ADServer.

During the installation, an AD called "ADExample" is also created which allows us to get a bit closer to the definition of AD and it will help us get to know the function of the ADServer.

Next we will describe the steps to be followed in more detail.

1. Run Tomcat.
2. Run the database MySQL.
3. Run a navigator and connect to http://localhost:catalina_port/ADServerSetup (catalina_port is the number of the open port to communicate with Tomcat). An installation window of the Active Document Server will show up (figure 3). In this window we will see a button called "Setup", which we will click on to begin the process of the installation of the ADServer.

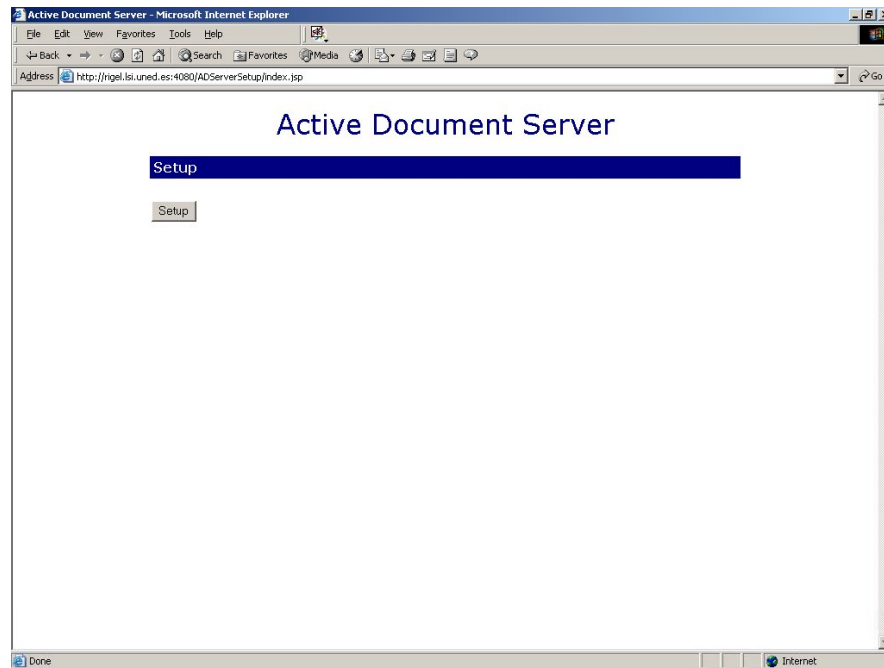


Figure 3. Accessing the ADServer installation

4. Configuration of the ADServer:
After clicking on the “Setup” button a window shows up (figure 4) where we are asked for information relative to the system that is necessary for the configuration of the ADServer.

Active Document Server - Microsoft Internet Explorer

Address <http://rigel.lsi.uned.es:4080/ADServerSetup/setup.jsp>

Setup Active Document Server

Step 1. General specifications

Host Name

Step 2. Java Virtual Machine Configuration

Java Home Path

Step 3. Web Server Configuration

Web Server Home Path

Port Number

Step 4. mySQL Database Manager Configuration

Port Number

User

Password

Step 5. Web Application Configuration

Context Name

Next Reset

Figure 4. Accessing the ADServer installation setup

This information is divided into five subsections. Next we will go on to discover exactly what each of the data asked for refers to.

General specifications		
Data relative to the identification of the machine		
Host Name	Explanation	It refers to the complete name of the machine in which we are installing the ADServer. It could be the IP address or the complete name.
	Value	Chain of text up to 64 characters. It cannot contain any blank spaces.
	Example	rigel.lsi.uned.es

Java Virtual Machine Configuration		
Information relative to the installation of the j2sdk		
Java Home Path	Explanation	Directory in which the j2sdk1.4 is installed.
	Value	It should be a direct folder route within the system.
	Example	C:\jdk1.4

Web Server Configuration		
Information relative to the installation of Tomcat		
Web Server Home Path	Explanation	Directory in which Tomcat is installed.
	Value	It should be a directory route which is valid within the system.
	Example	C:\tomcat
Port Number	Explanation	Port number through which the Tomcat web server flows.
	Value	It should be a numerical value that corresponds to a valid port.
	Example	4080

mySQL Database Manager Configuration		
Data relative to the BD MySQL to permit the ADServer the connection with the BD in order to work with the BDs of the results and log		
Port Number	Explanation	Port number through which the database mySQL will flow.
	Value	It should be a numerical value that corresponds to a valid port number.
	Example	3306
User	Explanation	The valid user in the MySQL database that needs permission to create databases and connect with the existing one in order to

		consult, add and modify information.
	Value	Chain of text with a length of up to 16 characters. It cannot contain any blank spaces.
	Example	ycalero
Password	Explanation	The last user's password for connection to the BD MySQL.
	Value	Chain of text with a length of up to 16 characters. It cannot contain blank spaces.
	Example	one_ex
IMPORTANT:	<p>If you don't have a user of this type, it must be signed up in the database of MySQL; to do that, we can enter in the monitor of BD and use the users' administration tool. The user will be user@hostname and should have permission to create and use the databases (as a creator of databases and charts, to consult and update or create indexes).</p> <p>If we also want the AD to work when connecting from any machine, we have to first sign up that user in order to be able to connect from any machine (this operation requires permission from the administrator):</p> <pre>INSERT INTO user (Host, User, Password) VALUES ('%', '<user>', PASSWORD('<password>')) ;</pre> <pre>GRANT INSERT, SELECT, UPDATE, DELETE ON *.* TO ycalero;</pre>	

Web Application Configuration		
Context Name	Explanation	The name that is desired to give to the context that is going to contain the Active Document Server.
	Value	Chain of text with a length of up to 16 characters. It cannot contain blank spaces.
	Example	ADServer

After filling in all fields (figure 5) click the button “Next”. At this moment the program will check to see if the information entered is correct and if it is, it proceeds to install the files that make up the ADServer.

Active Document Server - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Address <http://rigel.lsi.uned.es:4080/ADServerSetup/setup.jsp?> Go

Setup Active Document Server

Step 1. General specifications

Host Name

Step 2. Java Virtual Machine Configuration

Java Home Path

Step 3. Web Server Configuration

Web Server Home Path

Port Number

Step 4. mySQL Database Manager Configuration

Port Number

User

Password

Step 5. Web Application Configuration

Context Name

Done Internet

Figure 5. (Setting up) Configuring the ADServer installation on our systems

If an error is detected in the information entered, for example if the directories don't exist, or the connection to the database couldn't be carried out, an error message will show indicating what the problem was and there will be a “Correct” button which will allow you to go back to the previous window to correct it (figure 6).

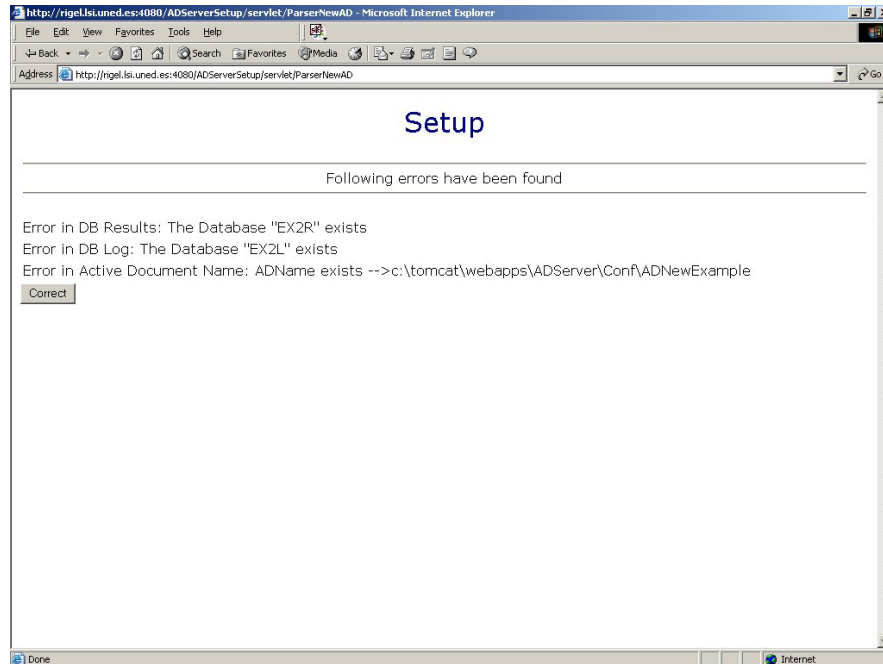


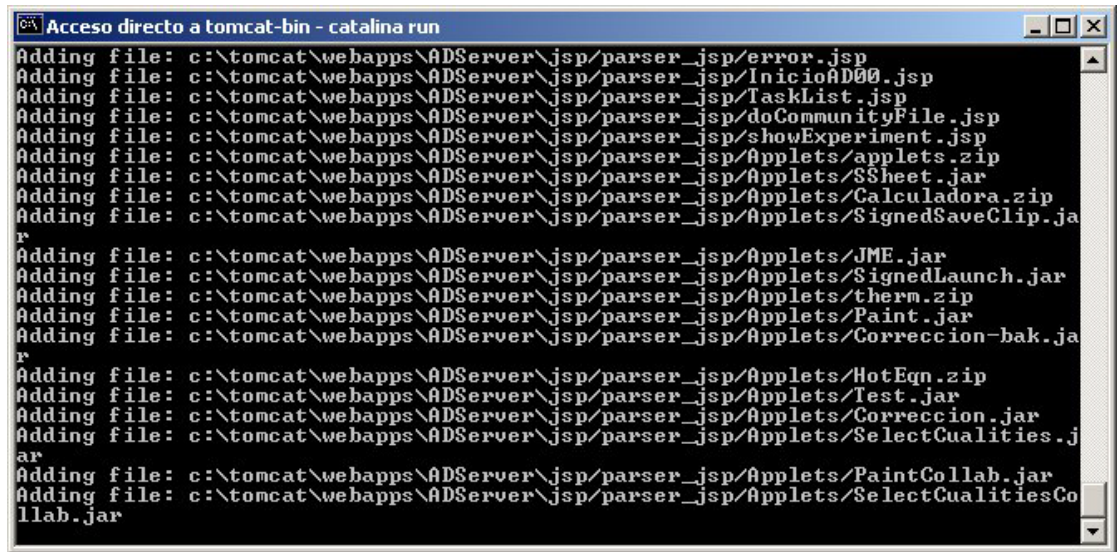
Figure 6. Error during the ADServer installation

5. Installation of the files that make up the ADServer:

If everything is correct, carry out the installation, which means:

- Copy the files that make up the ADServer
- Copy the files that make up the MetadataEditor
- Copy the files that make up the AD “ExampleAD”
- Create the databases of the results (DBResults) and of the log (DBLog) necessary for the functioning of the AD “ExampleAD”
- Create and update the configuration files

IMPORTANT: During the copying of the files you will be able to see what is happening in the Tomcat console (figure 7), in it we can see which files are being copied on to our machine.



```
Acceso directo a tomcat-bin - catalina run
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/error.jsp
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/InicioAD00.jsp
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/TaskList.jsp
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/doCommunityFile.jsp
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/showExperiment.jsp
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/applets.zip
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/SSheet.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/Calculadora.zip
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/SignedSaveClip.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/JME.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/SignedLaunch.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/therm.zip
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/Paint.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/Correccion-bak.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/HotEqn.zip
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/Test.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/Correccion.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/SelectCualities.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/PaintCollab.jar
Adding file: c:\tomcat\webapps\ADServer\jsp/parser_jsp/Applets/SelectCualitiesCollab.jar
```

Figure 7. Tomcat console

If any errors occur during the installation, a red message will appear indicating the place where it has occurred. In which case, since the installation hasn't been finished, we will have to see what produced it in the Tomcat console, try to solve it, uninstall the part of the ADServerSetup that has been installed (as described in section 2.2.7 *Un-installation of the ADServer*) and try again.

We know that the installation process has been carried out successfully if the following message appears (figure 8):

“The process has finished successfully”

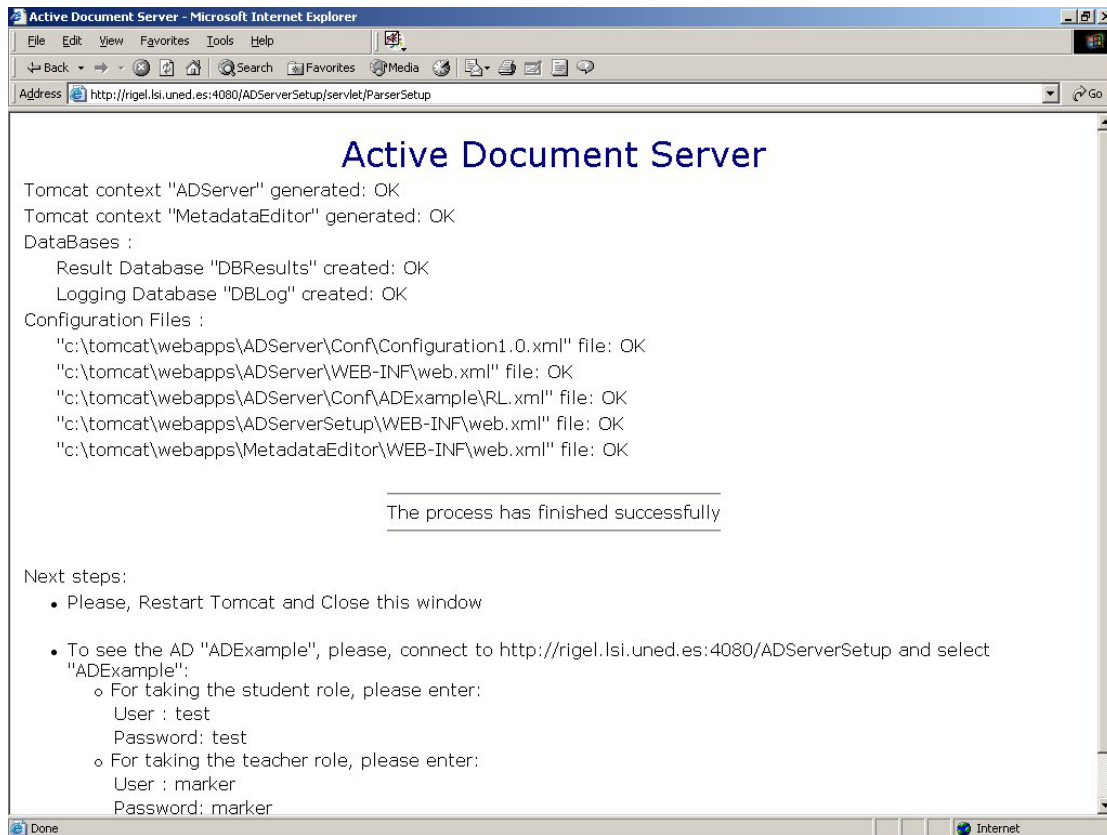


Figure 8. ADServer installation finished successfully

2.2.4 Updating after the changes

In order for the changes that were made to work, we have to re-start up Tomcat and close the navigator windows.

If we look in the Tomcat directory, we will see that two new contexts have been added: one belonging to the ADServer (it will have the name that was indicated in “Context Name”; see Web Application Configuration, p.13) and one called “MetadataEditor”.

The ADServer is installed with a demonstration AD called “AExample” so that the user has an example to refer to when creating their own Active Documents.

If we consult the databases defined in our system, we will see that two new databases have been created called “DBResult” and “DBLog” which are necessary for the AD to work “AExample”.

In the BD of results “DBResult” two users which are permitted to work with the AD are created by defect “AExample”:

To work in the role of student: Login=test Password=test

To work in the role of teacher: Login=marker Password=marker

2.2.5 Possible errors produced during installation

If any error in the information entered during the process of installation is detected or if any incoherence is detected, we might run into errors in the navigator window. Depending on the error, they are classified into the following groups:

2.2.5.1 Connection errors with the ADServerSetup

- ***The page cannot be displayed:*** If when connecting to the web page that allows us to enter the ADServer Setup we find a message like this, it means that Tomcat isn't active. The solution is to run Tomcat.

2.2.5.2 Format errors in the values entered by the user

- ***Error in [xxx]: The value is missing:*** No value was assigned to the installation parameter called "xxx". The solution is to assign a value to the parameter.
- ***Error in [xxx]: The value contains blanks:*** The value entered for the installation parameter "xxx" contains blank spaces. The solution is to eliminate the blank spaces that are in the value indicated in said parameter.
- ***Error in [xxx]: java.lang.NumberFormatException:*** This error means that the entered value must be a number. The solution is to indicate a number as the value.
- ***Error in [xxx]: [directory] Does not exists:*** The route indicated as a value for the installation parameter "xxx" doesn't exist, this means that the corresponding application to said parameter is not installed in the place indicated by the user. The solution is to check the installation route in the system and correct the value indicating the correct route.
- ***Error in [xxx]: [directory] Is not a directory:*** The route entered as value for the parameter of installation "xxx" is not a directory, which means the corresponding application to that parameter is not installed in the place indicated by the user. The solution is to check the installation route in the system and correct the value indicating the correct route.

2.2.5.3 Context errors provoked by previous installations

- ***Error in Context Name: Context exists --> [contextPath]:*** The Context Name chosen to install the ADServer already exists in Tomcat. This indicates that there is a context in Tomcat with the same name, which could be due to a previous installation of ADServer or that it coincides with another one that Tomcat is using. If it is because of a prior installation, un-install the ADServer as indicated in section 2.2.7 *Un-installation of the ADServer*, if not, indicate a different name in the installation parameter "Context Name".
- ***Error in MetadataEditor: metadataPath exists --> [metadataPath]:*** The context denominated "MetadataEditor" already exists within Tomcat. This is because the

ADServer has already been installed into the system; Re-start Apache. Un-install the ADServer as indicated in the section 2.2.7 *Un-installation of the ADServer*.

2.2.5.4 Errors with the DB

- **Error in [xxx]: The Database “[DB]” exists:** The database indicated as a value for the installation parameter “xxx” already exists. If it is because of the databases that are used in the example AD called DBLog and DBResults, this might be because the ADServer has already been installed into the system, un-in-install the ADServer as indicated in section 2.2.7 *Un-installation of the ADServer*. If it is a different DB, modify the value assigned to said installation parameter.
- **Error in connection DataBase: java.sql.SQLException: Cannot connect to MySQL server on [hostname:mysql_port]. Is there a MySQL server running on the machine/port you are trying to connect to? (java.net.ConnectException):** The system cannot connect to the database manager because it hasn’t been started up or because the port is not valid. Check if the hostname and the port number are valid and if the database is running in this host through this port; if they aren’t, run the database.
- **Error in connection DataBase: java.sql.SQLException: Invalid authorization specification: Access denied for user: '[user]@[hostname]’ (Using password: YES):** This indicates that the user or the password to make the connection to the DB are not valid or the user does not have permission to access the DB manager. Check that the user name corresponds to the defined user in the database and that the password is correct.

2.2.6 Configuration file

This file cannot be edited, rather it is generated during the installation of the ADServer and each time a new AD is added, the corresponding part to the configuration is added to the file. The configuration file fulfils the dtd called “Configuration.dtd”.

2.2.6.1 Elements for configuring the AD environment

AD

	<AD....>
Explanation	The files defining the AD elements
XML description	<!ELEMENT AD (URI_AD, URI_RL, URI_Community)>
Example	<AD> <URI_AD>c:\tomcat\webapps\ADServer\Conf\ADExample\AD.xml</URI_AD> <URI_RL>c:\tomcat\webapps\ADServer\Conf\ADExample\RL.xml</URI_RL> <URI_Community>c:\tomcat\webapps\ADServer\Conf\ADExample\CF.xml

	</URI_Community> </AD>
BinaryPathO	
	<BinaryPathO ...>
Explanation	URI for the binary objects that are to be included in the objects DB
XML description	<!ELEMENT BinaryPathO (#PCDATA)>
Example	Not available
BinaryPathR	
	<URI_RL ...>
Explanation	URI for the binary objects that are to be included in the results DB
XML description	<!ELEMENT BinaryPathR (#PCDATA)>
Example	<BinaryPathR>c:\tomcat\webapps\ADServer\Conf\ADExample\bin\</BinaryPathR>
CatalinaHome	
	<CatalinaHome...>
Explanation	Location of the Tomcat software on the AD system host
XML description	<!ELEMENT CatalinaHome (#PCDATA)>
Example	<CatalinaHome>c:\tomcat</CatalinaHome>
CatalinaPort	
	<CatalinaPort...>
Explanation	The port open on the host machine of the AD system for connecting to the Tomcat Web Server and Container
XML description	<!ELEMENT CatalinaPort (#PCDATA)>
Example	<CatalinaPort>4080</CatalinaPort>
Comment	
	<Comment ...>
Explanation	An optional comment that would appear on the right hand side of the AD heading
XML description	<!ELEMENT Comment (#PCDATA)>
Example	<Comment>Example</Comment>
Configuration	
Element	<Configuration...>
Explanation	The root element of the AD configuration
XML description	<!ELEMENT Configuration (ConfigurationSystem, GeneralConf, ConfigurationAD+)>

Example	See file configuration1.0.xml
ConfigurationAD	
	<ConfigurationAD....>
Explanation	Parameters for the AD configuration: the AD files, the Results DB, the Objects DB, the AD system logging configurarion, the XML results setup, the presentation parameters and the description and variables for the tools used by this AD. The required parameter is the AD unique identifier
XML description	<!ELEMENT ConfigurationAD (AD, DatabaseResult, DatabaseObject?, Log, ResultXML, Presentation, Tools?)> <!ATTLIST ConfigurationAD ADName ID #REQUIRED>
Example	See the examples for the parts composing this one
ConfigurationSystem	
	<ConfigurationSystem...>
Explanation	The elements defining the software installation for supporting the AD system
XML description	<!ELEMENT ConfigurationSystem (System, Tomcat, Mysql)>
Example	<pre> <ConfigurationSystem> <System> <OS>windows</OS> <Hostname>rigel.lsi.uned.es</Hostname> <JavaHome>c:\programas\jdk1.4</JavaHome> </System> <Tomcat> <CatalinaHome>c:\tomcat</CatalinaHome> <CatalinaPort>4080</CatalinaPort> </Tomcat> <Mysql> <jdbcDriver>org.gjt.mm.mysql.Driver</jdbcDriver> <mysqlPort>3306</mysqlPort> </Mysql> </ConfigurationSystem> </pre>
ContextName	
	<ContextName ...>
Explanation	The <i>context</i> according to Tomcat terminology. All the application is referred to that context
XML description	<!ELEMENT ContextName (#PCDATA)>
Example	<ContextName>ADServer</ContextName>
DataBaseLog	
	<DataBaseLog ...>
Explanation	The configuration elements which are necessary for logging onto a DB
XML description	<!ELEMENT DataBaseLog (DriverL, ServerL, NameL, UserL, PasswdL)>
Example	<pre> <DataBaseLog> <DriverL>org.gjt.mm.mysql.Driver</DriverL> <ServerL>jdbc:mysql://rigel.lsi.uned.es:3306</ServerL> <NameL>DBLog</NameL> </pre>

	<pre> <UserL>ycalero</UserL> <PasswdL>prueba</PasswdL> </DatabaseLog> </pre>
DatabaseObject	
	<DatabaseObject ...>
Explanation	The database for storing external objects, which would need the AD
XML description	<!ELEMENT DatabaseObject (DriverO, ServerO, NameO, UserO, PasswdO, BinaryPathO)>
Example	Not available
DatabaseResult	
	<URI_DatabaseResult...>
Explanation	Parameters for setting up the AD results DB
XML description	<!ELEMENT DatabaseResult (DriverR, ServerR, NameR, UserR, PasswdR, BinaryPathR)>
Example	<pre> <DatabaseResult> <DriverR>org.gjt.mm.mysql.Driver</DriverR> <ServerR>jdbc:mysql://rigel.lsi.uned.es:3306</ServerR> <NameR>DBResults</NameR> <UserR>ycalero</UserR> <PasswdR>prueba</PasswdR> <BinaryPathR>c:\tomcat\webapps\ADServer\Conf\ADExample\bin\</BinaryPathR> </DatabaseResult> </pre>
DirResult	
	<DirResult ...>
Explanation	The location (directory) of the XML results
XML description	<!ELEMENT DirResult (#PCDATA)>
Example	<DirResult>c:\tomcat\webapps\ADServer\Conf\ADExample\Results\</DirResult>
DriverO	
	<DriverO ...>
Explanation	Driver for connecting to the objects DB
XML description	<!ELEMENT DriverO (#PCDATA)>
Example	Not available
DriverR	
	<DriverR ...>
Explanation	JDBC driver for connecting to the results DB
XML description	<!ELEMENT DriverR (#PCDATA)>
Example	<DriverR>org.gjt.mm.mysql.Driver</DriverR>
GeneralConf	

	<GeneralConf...>
Explanation	Common configuration elements shared by every AD defined within this AD system
XML description	<!ELEMENT GeneralConf (ContextName, URI_ADServlet, URI_BaseApplet)>
Example	<pre> <GeneralConf> <ContextName>ADServer</ContextName> <URI_ADServlet>http://rigel.lsi.uned.es:4080/ADServer/servlet/ADServlet </URI_ADServlet> <URI_BaseApplet>http://rigel.lsi.uned.es:4080/ADServer/jsp/parser_jsp/Applets </URI_BaseApplet> </GeneralConf> </pre>
Hostname	
	<hostname...>
Explanation	Fully qualified name (but not URL) of the AD system host machine
XML description	<!ELEMENT Hostname (#PCDATA)>
Example	<Hostname>rigel.lsi.uned.es</Hostname>
Icon	
	<Icon ...>
Explanation	This icon will appear at the bottom of the displayed AD
XML description	<!ELEMENT Icon (#PCDATA)>
Example	<Icon>http://rigel.lsi.uned.es:4080/ADServer/Conf/ADEExample/Presentation/Divilab.jpg</Icon>
JavaHome	
	<JavaHome...>
Explanation	The location of the Java Virtual Machine on the AD system host
XML description	<!ELEMENT JavaHome (#PCDATA)>
Example	<JavaHome>c:\programas\jdk1.4</JavaHome>
JdbcDriver	
	<jdbcDriver...>
Explanation	Fully qualified name of the Java DB Connection Driver for the MySQL DB
XML description	<!ELEMENT jdbcDriver (#PCDATA)>
Example	<jdbcDriver>org.gjt.mm.mysql.Driver</jdbcDriver>
Log	
	<Log ...>
Explanation	<p>The configuration elements for the logging system. It consists of a DBLogging and a XMLLogging elements</p> <p><u>Attributes:</u></p>

	<ul style="list-style-type: none"> ➤ XML: recording or not in XML format ➤ DB: recording or not in DB format
XML description	<pre><!ELEMENT Log (DatabaseLog, LogXML)> <!ATTLIST Log XML (yes no) #REQUIRED DB (yes no) #REQUIRED></pre>
Example	<pre><Log XML="no" DB="no"> <DatabaseLog> <DriverL>org.gjt.mm.mysql.Driver</DriverL> <ServerL>jdbc:mysql://rigel.lsi.uned.es:3306</ServerL> <NameL>DBLog</NameL> <UserL>ycalero</UserL> <PasswdL>prueba</PasswdL> </DatabaseLog> <LogXML> <UriDTDLog>http://rigel.lsi.uned.es:4080/ADServer/Conf/AD_Log_v1.0.dtd </UriDTDLog> <PathLog>c:\tomcat\webapps\ADServer\Conf\ADExample\Log\log.xml</PathLog> </LogXML> </Log></pre>
LogXML	
	<LogXML ...>
Explanation	The required setup for XML logging
XML description	<pre><!ELEMENT LogXML (UriDTDLog, PathLog)></pre>
Example	<pre><LogXML> <UriDTDLog>http://rigel.lsi.uned.es:4080/ADServer/Conf/AD_Log_v1.0.dtd </UriDTDLog> <PathLog>c:\tomcat\webapps\ADServer\Conf\ADExample\Log\log.xml</PathLog> </LogXML></pre>
Menu	
	<Menu ...>
Explanation	There can be any number of menus (even none)
XML description	<pre><!ELEMENT Menu (MenuOption+)></pre>
Example	<pre><Menu> <MenuOption uri="http://www.uned.es" name="Uned"/> </Menu></pre>
MenuOption	
	<MenuOption ...>
Explanation	<p>Each of the available options which would compose the current menu</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - Name: the menu title - URI: the URI to go when the user selects this option
XML description	<pre><!ELEMENT MenuOption EMPTY> <!ATTLIST MenuOption</pre>

	name CDATA #REQUIRED uri CDATA #REQUIRED
Example	<MenuOption uri="http://www.uned.es" name="Uned"/>
MySQL	
	<Mysql...>
Explanation	Elements necessary for configuring the MySQL DataBase application and server
XML description	<!ELEMENT Mysql (jdbcDriver, mysqlPort)>
Example	<Mysql> <jdbcDriver>org.gjt.mm.mysql.Driver</jdbcDriver> <mysqlPort>3306</mysqlPort> </Mysql>
MysqlPort	
	<mysqlPort...>
Explanation	Port open on the host machine of the AD system for connecting to the MySQL DB server
XML description	<!ELEMENT mysqlPort (#PCDATA)>
Example	<mysqlPort>3306</mysqlPort>
NameO	
	<NameO ...>
Explanation	Name of the objects DB
XML description	<!ELEMENT NameO (#PCDATA)>
Example	Not available
NameR	
	<NameR ...>
Explanation	Results DB name
XML description	<!ELEMENT NameR (#PCDATA)>
Example	<NameR>DBResults</NameR>
OS	
	<os...>
Explanation	The operating system running on the AD system host
XML description	<!ELEMENT OS (#PCDATA)>
Example	<OS>windows</OS>
ParamConf	
	<ParamConf ...>
Explanation	Any of the parameters to be supplied to the tool being defined <u>Attributes:</u>

	<ul style="list-style-type: none"> ➤ paramAD: AD internal parameter identifier after which will be valued the tool's parameter ➤ paramTool: tool's parameter to be valued after its AD counterpart
XML description	<pre><!ELEMENT ParamConf EMPTY> <!--ATTLIST ParamConf paramAD CDATA #REQUIRED paramTool CDATA #REQUIRED--></pre>
Example	<ParamConf paramTool="marker" paramAD="login"/>
PasswdO	
	<PasswdO ...>
Explanation	Password for the user identified by <userO> to connect to the objects DB
XML description	<!ELEMENT PasswdO (#PCDATA)>
Example	Not available
PasswdR	
	<PasswdR ...>
Explanation	Password for the user defined as <UserR> to connect to the results DB
XML description	<!ELEMENT PasswdR (#PCDATA)>
Example	<PasswdR>prueba</PasswdR>
Editor window	<i>Not available</i>
PathLog	
	<PathLog...>
Explanation	The logging file path
XML description	<!ELEMENT PathLog (#PCDATA)>
Example	<PathLog>c:\tomcat\webapps\ADServer\Conf\ADExample\Log\log.xml</PathLog>
Presentation	
	<Presentation ...>
Explanation	The elements necessary for customising the AD appearance (how it will be displayed)
XML description	<!ELEMENT Presentation (Title, Comment, Icon, Style, Menu?)>
Example	<pre><Presentation> <Title>Active Document</Title> <Comment>Example</Comment> <Icon>http://rigel.lsi.uned.es:4080/ADServer/Conf/ADExample/Presentation/Divilab.jpg </Icon> <Style>http://rigel.lsi.uned.es:4080/ADServer/Conf/ADExample/Presentation/ActiveDocument.css</Style> <Menu></pre>

	<pre> <MenuOption uri="http://www.uned.es" name="Uned"/> </Menu> </Presentation> </pre>
ResultXML	
	<ResultXML ...>
Explanation	The elements necessary for setting up the XML results
XML description	<!ELEMENT ResultXML (UriDTDResult, DirResult)>
Example	<pre> <ResultXML> <UriDTDResult>http://rigel.lsi.uned.es:4080/ADServer/Conf/AD_growable_v3.dtd </UriDTDResult> <DirResult>c:\tomcat\webapps\ADServer\Conf\ADExample\Results\</DirResult> </ResultXML> </pre>
ServerO	
	<ServerO ...>
Explanation	URL of the server for connecting to the objects DB
XML description	<!ELEMENT ServerO (#PCDATA)>
Example	Not available
ServerR	
	<ServerR ...>
Explanation	URL for the server allowing the connection to the results DB
XML description	<!ELEMENT ServerR (#PCDATA)>
Example	<ServerR>jdbc:mysql://rigel.lsi.uned.es:3306</ServerR>
Style	
	<Style ...>
Explanation	Style sheet to be applied to the AD presentation
XML description	<!ELEMENT Style (#PCDATA)>
Example	<Style>http://rigel.lsi.uned.es:4080/ADServer/Conf/ADExample/Presentation/ActiveDocument.css</Style>
System	
	<system...>
Explanation	The description of the basic system elements, i.e., operating system under which the AD system is to be run, name of the AD system host and location of the Java VM
XML description	<!ELEMENT System (OS, Hostname, JavaHome)>
Example	<pre> <System> <OS>windows</OS> <Hostname>rigel.lsi.uned.es</Hostname> <JavaHome>c:\programas\jdk1.4</JavaHome> </pre>

	</System>
Title	
	<Title ...>
Explanation	This title will be shown on the AD heading
XML description	<!ELEMENT Title (#PCDATA)>
Example	<Title>Active Document</Title>
Tomcat	
	<Tomcat...>
Explanation	Elements necessary for configuring the Tomcat Web Server and Application Container (for further information, see the Apache Jakarta Project at the URL: http://jakarta.apache.org/tomcat/)
XML description	<!ELEMENT Tomcat (CatalinaHome, CatalinaPort)>
Example	<pre> <Tomcat> <CatalinaHome>c:\tomcat</CatalinaHome> <CatalinaPort>4080</CatalinaPort> </Tomcat> </pre>
ToolConf	
	<ToolConf ...>
Explanation	<p>Any of the tool configuration entries</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - id: the tool's unique identifier - name: a name for the tool, other than its id
XML description	<pre> <!ELEMENT ToolConf (ParamConf+)> <!ATTLIST ToolConf id CDATA #REQUIRED name CDATA #IMPLIED> </pre>
Example	<pre> <ToolConf id="correction"> <ParamConf paramTool="marker" paramAD="login"/> <ParamConf paramTool="student" paramAD="student"/> <ParamConf paramTool="labdoc" paramAD="labdoc"/> <ParamConf paramTool="experience" paramAD="experiment"/> <ParamConf paramTool="activity" paramAD="activity"/> <ParamConf paramTool="task" paramAD="taskStu"/> <ParamConf paramTool="host" paramAD="ServerDBR"/> </ToolConf> </pre>
Tools	
	<Tools ...>
Explanation	This element is used for setting up the internal ¹ tools the AD would be using. These tools will be needing AD values to work with. The connection for passing values to the tools and back will be made through a servlet
XML	<!ELEMENT Tools (ToolConf+)>

¹ Internal tools refer to tools that can only be used with the AD System. Conversely, external tools are standalone applications that have been configured to work with the AD System.

description	
Example	<pre> <Tools> <ToolConf id="DrawTool"> ... stuff cut for saving space see the following elements for details ... </ToolConf> <ToolConf id="tooltest"> ... stuff cut for saving space see the following elements for details ... </ToolConf> <ToolConf id="correction"> ... stuff cut for saving space see the following elements for details ... </ToolConf> </Tools> </pre>
URI_AD	
	<URI_AD....>
Explanation	URI for the AD definition file
XML description	<!ELEMENT URI_AD (#PCDATA)>
Example	<URI_AD>c:\tomcat\webapps\ADServer\Conf\ADExample\AD.xml</URI_AD>
URI_ADServlet	
	<URI_ADServlet ...>
Explanation	Data base connection servlet. Every external tool will be connected through this servlet
XML description	<!ELEMENT URI_ADServlet (#PCDATA)>
Example	<URI_ADServlet>http://rigel.lsi.uned.es:4080/ADServer/servlet/ADServlet</URI_ADServlet>
URI_BaseApplet	
	<URI_BaseApplet>
Explanation	URI for the applets that the AD will be using internally for its operation
XML description	<!ELEMENT URI_BaseApplet (#PCDATA)>
Example	<URI_BaseApplet>http://rigel.lsi.uned.es:4080/ADServer/jsp/parser_jsp/Applets</URI_BaseApplet>
URI_Community	
	<URI_Community ...>
Explanation	URI for the AD community definition file
XML description	<!ELEMENT URI_Community (#PCDATA)>
Example	<URI_Community>c:\tomcat\webapps\ADServer\Conf\ADExample\CF.xml</URI_Community>
URI_RL	
	<URI_RL ...>

Explanation	URI for the AD resource list file
XML description	<!ELEMENT URI_RL (#PCDATA)>
Example	<URI_RL>c:\tomcat\webapps\ADServer\Conf\ADExample\RL.xml</URI_RL>
UriDTDLog	
	<UriDTDLog ...>
Explanation	The URI of the DTD for the log files
XML description	<!ELEMENT UriDTDLog (#PCDATA)>
Example	<UriDTDLog>http://rigel.lsi.uned.es:4080/ADServer/Conf/AD_Log_v1.0.dtd</UriDTDLog>
UriDTDResult	
	<UriDTDResult...>
Explanation	The URI of the XML results DTD
XML description	<!ELEMENT UriDTDResult (#PCDATA)>
Example	<UriDTDResult>http://rigel.lsi.uned.es:4080/ADServer/Conf/AD_growable_v3.dtd</UriDTDResult>
UserO	
	<UserO ...>
Explanation	The user identifier for connecting to the objects DB.
XML description	<!ELEMENT UserO (#PCDATA)>
Example	Not available
UserR	
	<UserR ...>
Explanation	User identifier for connecting to the results DB. This user needs DB, table and indices creating, selecting and updating permissions on the results DB, and such a user must exist for the proper configuration and operation of the AD
XML description	<!ELEMENT UserR (#PCDATA)>
Example	<UserR>ycalero</UserR>

2.2.7 Un-installation of the ADServer

To un-install the system, you must follow these steps:

Within the webapps directory of Tomcat:

- Eliminate the following directories:
 - ADServer (or the name that was entered as “Context Name”)

- MetadataEditor
- Within the directory ADServerSetup\WEB-INF edit the file called web.xml and erase the following part :


```
<context-param>
  <param-name>configFile</param-name>
  <param-value>c:\tomcat\webapps\ADServer\Conf\Configuration1.0.xml</param-value>
  <description>
    Complete file path to the configuration file de desarrollo
  </description>
</context-param>
```
- Using a database manager, eliminate the databases used for the example AD that includes the installation program, these are denominated: DBLog and DBResults.

Then close the navigator windows and re-start Apache.

Once all of this has been done, we can install the AD again, following the steps in section 2.2.3 *Running the installation program*.

2.3 Installing the Task Manager

The installation process is composed of the following seven steps:

- 1) Unzip the distribution tm.zip in a folder.
- 2) Create a database for the task manager.

Ask your system administrator to create a database and ask him the following information:

- the database java url to join the database : It should be something like :
jdbc:postgresql: //hostname:port/tm?charSet=LATIN1 for a database on postgresql whose name is tm
- an admin login which has rights to create tables in the database
- the admin password
- a user login which is able to execute all the current sql statements on the database (select, insert...)
- the user password

On your side, verify you have got the correct database jar file to connect on the database and note the jdbc driver name. If you need to use another database jar file, please copy it into the tm.0.4.1_00/tm/WEB-INF/lib folder and sets this information into the build properties file.

- 3) Modify the build.properties file.

If you want to create database tables with the tool provided by task manager, please edit the build.properties file to set the following information:

- **db.driver**=The jdbc driver name
- **db.url**=the database java url
- **db.login**=The admin login
- **db.password**=the admin password
- **driver.jar.name**=the database jar file name
- **db.filename** which locates the file which stores the sql requirements useful to build the tables in the database. It should be equals to properties/postgresql.sql or properties/mysql.sql

In any case set the following information:

- **catalina.home**=the path to Tomcat home directory

4) Modify the web.xml file

Edit the web.xml file to set the following information :

- **tmDbDriver** : The jdbc drivename
- **tmDbUrl** : the database java url
- **tmDbLogin** : The user login
- **tmDbPassword** : the user password

5) Create the tables in the database

- Either open a window in the tm0.4.1_00 folder and run **ant dbload** to create the tables in your database and verify that the last line written by the tool says that there is no error
- Either execute the postgresql.sql or mysql file with the tool provided by the database manager

6) Deploy the task manager application

Open a window in the tm0.4.1_00 folder and execute ant deploy

7) Run the Task manager

Start (or restart) Tomcat and then check that the Task Manager has been correctly installed by connecting <http://hostname:8080/tm/test.htm> from a Web browser., (where hostname:8080 represents the hostname and port of tomcat server). This page regroups a set of sample commands useful to understand how the task manager interacts with a client application.

3 Installation of an AD client

3.1 System requirements

To connect with the ServerAD as a client, a PC with the following characteristics must be available:

- Windows 98 / Me / 2000 / XP
- 64MB of memory minimum
- A screen resolution of 800x600 or 1024x768

We need to have a series of programs installed in our PC in order for it to work:

- Navigator:
 - Microsoft Internet Explorer 5 or higher
 - Netscape Navigator 6.2
- Java plugin: j2re_1_4_0
- SVG plugin : SVGView

3.2 Installing the program

In order for it to work, we must have a series of programs installed in our PC that if they are not currently installed, they must be installed. Next, we will see the programs necessary and the instructions to install them.

3.2.1 Navigator

In order to be able to use this system, you must have one of the following web navigators installed:

- Microsoft Internet Explorer 5 or higher
- Netscape Navigator 6.2

You can check the version in the following way:

- On Microsoft Internet Explorer, in the help menu, click on “About Internet Explorer”
- On Netscape Navigator, in the menu Help, click on “About Netscape”

If you do not have the version indicated above, it must be installed with the files provided on this CDROM (version 6 of Microsoft Internet Explorer or version 6.2 of

Netscape Navigator for Windows). **The use of Microsoft Internet Explorer is recommended.**

To install Netscape Navigator 6, you have to open the folder Netscape 6.2 in the CDROM, double click on the file N6SetupB and follow the directions.

To install Microsoft Internet Explorer, the folder Iexplorer6 must be opened in the CDROM, double click on the file ie6setup and follow the directions.

3.2.2 Java plugin

After having installed the navigator, the Java plugin (program) that is found in the CDROM, double click on the file j2re-1_4_0-win and follow the directions. In this installation program you must select the Web navigators that you have installed in your machine, as shown in figure 9.

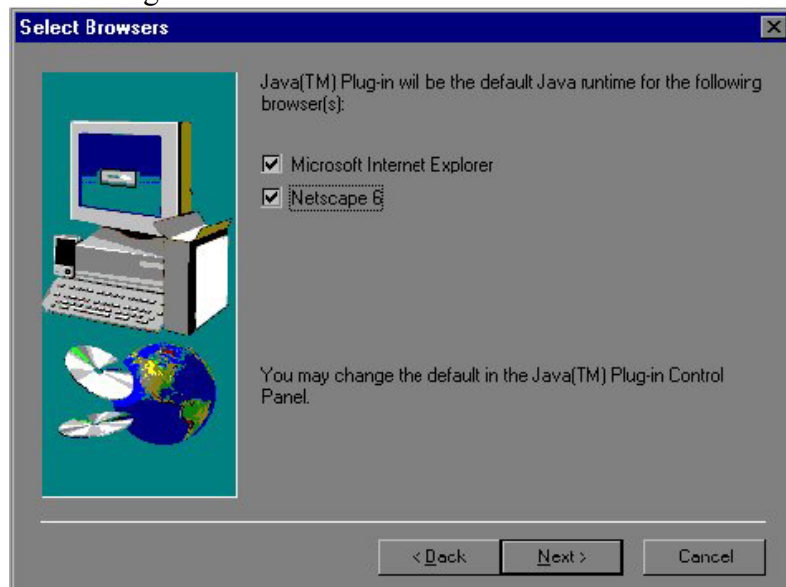


Figure 9. Java plugin installation

3.2.3 SVG plugin

After installation, you must install the SVG plugin that is on the CDROM, double click on the file SVGView and follow the directions.

3.3 Creation of a new AD

To create a new AD, we must open a navigator and connect to: http://localhost:catalina_port/ADServerSetup

To create a new AD we click on the button “NewAD” and a window will appear where it asks for the necessary information to create a new AD (figure 10).

New AD

Step 1. MySQL Database Manager Configuration

User

Password

Step 2. Web Application Configuration

General

Active Document

DataBases

DBResults

DBLog

Logging

Database

XML

Presentation

Title

Comment

Icon

Stylesheet

Menu

Tools

Figure 10. Creating a new AD

This information is divided into various sections that are described below:

MySQL Database Manager Configuration (Step 1)

Data relative to the DB MySQL to allow the ADServer the connection with the DB in order to be able to work with the DB results and log

User	Explanation	Valid user in the database MySQL which allows the creation of databases and the connection to an already existing one in order to consult, add and modify information.
	Value	A Chain of text with a length of 16 characters. It cannot contain blank spaces.
	Example	ycalero
Password	Explanation	Previous user password for connection to the DB mySQL.
	Value	A Chain of text with a length of 16 characters. It cannot contain blank spaces.
	Example	one_ex

Web Application Configuration → General (Step 2)		
Data relative to the new AD		
Active Document	Explanation	Active Document identifier. It cannot contain blank spaces.
	Value	A Chain of text with a length of 16 characters. It cannot contain blank spaces.
	Example	ADNewExample

Web Application Configuration → DataBases (Step 2)		
Solicits the names of the resulting DB and log that will be created along with the AD. These DB are necessary for the functioning of the AD.		
DBResults	Explanation	The name of the database that is going to be used in this AD to store the resulting tasks.
	Value	A Chain of text with a length of 16 characters. It cannot contain blank spaces.
	Example	EX2R

DBLog	Explanation	Name of the database that is going to be used to store the access log in the database in the case that the log is to be stored in the DB.
	Value	A Chain of text with a length of 16 characters. It cannot contain blank spaces.
	Example	EX2L

Web Application Configuration → Logging (Step 2)		
Allows the user to define whether or not to store the log information relative to the use of the AD		
Database	Explanation	Indicates if it is desired to store the AD access log in the database.
	Value	The value can be yes or no.
	Example	no
XML	Explanation	Indicates if it is desired to store the ADaccess log as a XML file.
	Value	The value can be yes or no.
	Example	no

Web Application Configuration → Presentation (Step 2)		
It permits defining the appearance that the AD will have (in order to better understand what each solicited value refers to, see figure 86)		
Title	Explanation	Title that appears at the head of the active document.
	Value	Chain of text with a length of up to 64 characters.
	Example	Active Document
Comment	Explanation	Comment that appears on the right hand side of the header of the active document.
	Value	Chain of text with a length of up to 64 characters.
	Example	Comment

Icon	Explanation	Icon that appears in the lower part of the AD
	Value	Value by defect: ltes.gif It must be the name of a valid image file. If the value by defect is modified, after creating the new AD, the user must copy said file in the directory: catalinaHome/catalina_port/context_name/ad_name/Presentation/
	Example	Divilab.jpg
Stylesheet	Explanation	Stylesheet that is applied to the AD
	Value	Value by defect: ActiveDocument.css It must be the name of a valid style file. If the value by defect has been modified, after creating the new AD, the user must copy said file in the directory: catalinaHome/catalina_port/context_name/ad_name/Presentation/
	Example	ActiveDocument.css
IMPORTANT	<p>After creating the new AD, the files that the Icon refers to and the style that appear by defect can be found in the directory:</p> <p>catalinaHome/catalina_port/context_name/ad_name/Presentation/</p> <p>If the values that are offered by defect are modified, when the creation of the new AD is complete, the user will have to copy the files in this directory.</p>	

Web Application Configuration→Presentation→Menu(Step 2)		
There can be from 0 to n menus.		
Name	Explanation	Title of the menu that appears in the AD
	Value	Chain of text with a length of up to 32 characters.
	Example	UNED

URL	Explanation	URL that you will go to if you click on menu
	Value	It must be a valid URL. It cannot contain blank spaces.
	Example	http://www.uned.es

Web Application Configuration→Presentation→Tools(Step 2)

There can be from 0 to n defined tools.

It is used to configure the AD's internal tools that need values from the AD.

Identifier	Explanation	Tool identifier
	Value	Chain of text with a length of up to 64 characters. It cannot contain blank spaces. It is the ID of a tool defined within the resource folder.
	Example	DrawTool
IMPORTANT:	<p>This part is only necessary if internal tools are added to the ad.</p> <p>The AD that are created are installed by defect with the following internal tools:</p> <ul style="list-style-type: none"> ➤ Editor ➤ DrawTool ➤ Correction_Tool 	

Web Application Configuration→ Presentation→Tools→ Parameters

An internal tool that needs internal AD values can have from 1 to n parameters.

This allows us to specify what parameter tool will have a value from the AD.

ActiveDocument Parameter	Explanation	Identifier of the internal AD parameter that will be used to give value to the "Parameter Tool"
	Value	The parameters defined internally in the AD that can ADServletURL, ADName, community; login; idResul, view, login, labdoc, experiment, activity, task, result, idResul, members, userno.
	Example	login
Parameter Tool	Explanation	Name of the parameter tool that will take the value of the internal AD parameter specified in "ActiveDocument Parameter"
	Value	Chain of text with a length of up to 32 characters. It cannot

		contain blank spaces. It must be the name of a parameter defined in said tool.
	Example	login
<i>IMPORTANT:</i>	This part is only necessary if internal tools are added to the AD. The ADs that are created are installed with the following internal tools: <ul style="list-style-type: none"> ➤ Editor ➤ DrawTool ➤ CorrectionTool 	

After filling in all fields (figure 11), click the button “Next”. At that time the program will check if the information entered is correct and if it is, it will proceed with the installation of the files that make up the new AD.

New AD

Step 1. mySQL Database Manager Configuration

User:

Password:

Step 2. Web Application Configuration

General

Active Document:

DataBases

DBResults:

DBLog:

Logging

Database:

XML:

Presentation

Title:

Comment:

Icon:

Stylesheet:

Menu

Name:

URL:

Tools

Figure 11. Configuring a new AD

If any errors are detected in the information entered, such as the databases already existing, the connection to the DB cannot be carried out, the AD identifier exists, etc, you will see an error message indicating what the problem is and there will be button that says “Correct” which will allow you to go back to the previous window to correct the problem (figure 12).

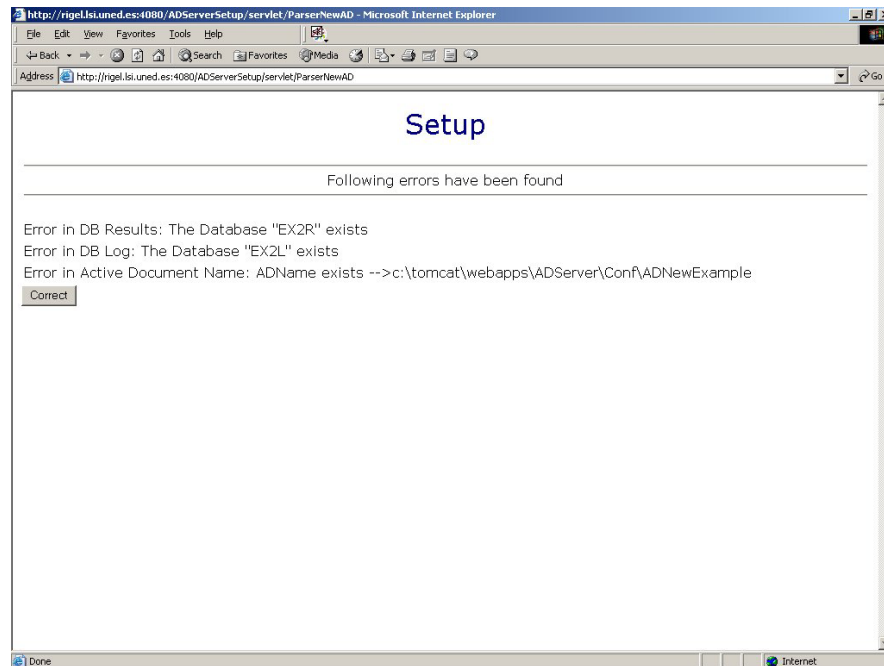


Figure 12. Error creating a new AD

If everything is correct, a new AD will be created which will consist of:

- Copying the files that make up the new AD
- Creating DB results and log necessary to work
- Creating and updating the configuration files.

If any errors occur during the creation of the new AD, a red error message will appear indicating at which point the error was produced. In that case, since the new AD will not be complete, we will have to find out what produced it in the Tomcat console, try to solve it, un-install the part of the new AD that has been installed (as described in section 4.4 *Eliminating an AD* and try again.

We will know the installation process has been completed correctly when we see the following message (figure 13).

“The process has finished successfully”

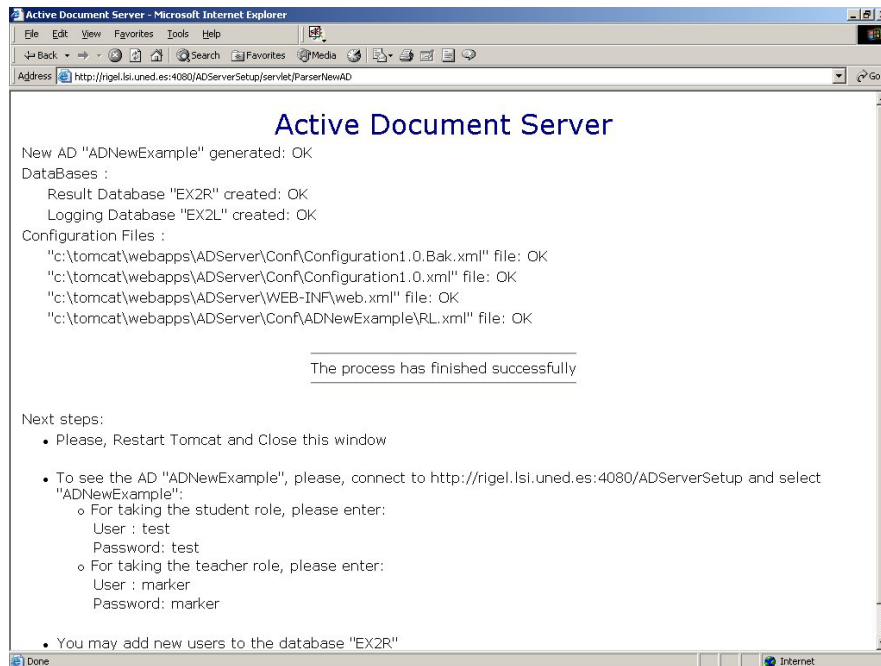


Figure 13. New AD created successfully

When an AD is created, by defect two users are registered that are allowed to work with the AD (we can see them in the results database, under the table “users”):

To work in the role of a student:

Login: test

Password: test

To work in the role of a teacher:

Login: marker

Password: marker

If we need to allow the access to an AD to different users than these, all we have to do is, using the manager BD of MySQL, connect to the results database that this AD is working with and insert lines in the table “users” of the following type:

INSERT INTO users

VALUES("<id>", "<login>", "<password>", "<name>", "<type>", "", "", "", "");

Where:

- <id>: User identification number.
- <login>: User name used to access the system.
- <password>: Password used to access the system
- <name>: Full user name

- <type>: User type

3.3.1 Errors when a new AD is created

When a new AD is created, apart from the errors indicated in section 2.2.5 *Possible errors produced during installation* the following errors can also be produced:

- **Error in Active Document Name: ADName exists -> [xxx]:** The identifier chosen for the new Active Document already exists in the ADServer. Choose a different identifier.
- **Error in some Menu Name: The value is missing:** The title in some menu has not been indicated. Indicate a name for the menu.
- **Error in some Menu Url: The value is missing:** The URL in some menu has not been indicated. Indicate a URL for the menu.
- **Error in some Menu Url: The value of [menuUri] contains blanks:** The value of the Uri indicated in [menuUri] has blank spaces. Eliminate the blank spaces that are contained in the URL.
- **Error in some Identifier: The value of Id is missing:** The value of the tool identifier has not been indicated. Identify a value for the tool identifier.
- **Error in some Identifier: The value of Id Tool "[id]" contains blanks:** The value indicated as tool identifier has blank spaces. Eliminate the blank spaces that are in the tool identifier.
- **Error in some Identifier: The value of Id is not valid:** The value indicated as tool identifier is not valid. Change the value of this ID for one that is valid.
- **Error in some Parameter: The value of parameter in Id Tool [id] is missing:** The parameter value has not been indicated. Indicate the value for the parameter of the AD.
- **Error in some Identifier: Id Tool have not parameters:** The tool parameters have not been indicated. Indicate a value for the tool parameter.
- **Error in some Parameter: The value of parameter AD [ParamAD] contains blanks:** The value indicated as AD parameter contains blank spaces. Eliminate the blank spaces contained in the AD parameter.
- **Error in some Parameter: The value of parameter Tool [ParamT] contains blanks:** The value indicated as parameter tool has blank spaces. Eliminate the blank spaces that are contained in the parameter tool.

3.4 Description of the AD XML files

The concept of an AD includes the following aspects of the learning design process: a description of the activities, a description of the communities, the resources available, and the outcome of the work undertaken within the environment. The ADs are specified in XML and are defined by four pairs of document type definitions (or DTDs) and their

corresponding XML document. The ADs are: (1) the description of the division of labour in the tasks and subtasks (referred to as the 'Description AD'). (2) The actors and roles involved in the collaborative tasks (referred to as the 'Community AD'); (3) Definition of the resources for carrying out the tasks and (4) the outcome of the activity (referred to as the 'Outcome AD'). The Description AD, along with the specification of the actors that perform the collaborative activities (specified in the Community AD) are interpreted by the AD architecture that dynamically creates the appropriate user interface, according to the elements defined in the two XML structures. As the learning activity proceeds, the outcome produced by each student is represented in XML in the Outcome AD which stores the results of the learning process and the task structure described in the Description AD. These ADs are described next.

- **The Description AD**

The 'Description AD' specifies a collection of activities, each of which reflect the components of an activity as described by Activity Theory, modelling the division of labour and the mediating tools associated with each task. Activities can be grouped within this AD, to provide (optional) sequencing and prerequisite dependences between *groups* of activities. The definition of an activity includes the following: (a) The description of the object of the activity. (b) The specification of the tasks and subtasks, and for each one (if applicable) the different roles that the participants involved in the task can play and (c) the instances (with the input/output data references) of tools and resources available for each role related to a task.

- **The Community Definition**

These components are also expressed in XML in a similar way as an activity. **The Community AD** represents the activity organization in order to describe the assignment of roles for a specific task to the members of a given community. For each activity, a description of the community involved is provided. As has been previously stated, this description is processed by the AD architecture in combination with the Description AD in order to relate the appropriate tasks and tools to the corresponding members of the community. The use of a separate XML structure for the community gives rise to two interesting mechanisms: firstly, communities can change during the development of the activity, thus allowing dynamic role assignments to be made (amongst other possibilities); and secondly, different Community AD can be combined with the same Description AD, providing a flexible mechanism for the re-use of the same division of labour description for a set of different working groups.

Note that in the context of DiViLab, this file is automatically generated by the Archimed LMS, as will be detailed in deliverable D7.7.

- **The resource definition**

The Description AD specifies the resources that are used in a scenario. The resources can range from simple textual documents to fully distributed applications. Tools can be invoked locally or remotely, and can be used synchronously and asynchronously. The resource definition specifies the way in which an instantiation of a resource handles its inputs and outputs in a form that is both coherent in the context of the overall AD system and transparent to the user. This transparency is a first step toward functional interoperability.

- **The outcome Definition**

The Outcome AD specifies the way in which the results of the tasks performed in the environment are stored, thus providing an *active* component, a vision of the current work completed and in-progress. Thus, the Outcome AD is in fact the real *active* component of the AD organization, i.e., it is the result of the work generated by a specific actor involved in the activity described by the Description AD. This representation provides a definition, at the desired level of detail, of the work and the objects generated during the learning process. The Outcome AD, rather than a sequence of plain text can contain complex elements like graphics, tables, structured dialogs, maps, etc., in an XML format embedded into it, with links to non XML objects outside, e.g., a MS Word document. Furthermore, the AD architecture makes this structured collection of heterogeneous objects *persistent* during the life cycle of the user within the environment, providing tools for their manipulation, storage and retrieval. This mechanism forms the basis for passing objects between tools in a transparent way for the user. Some interesting applications can be considered due to the nature of the Outcome AD representation. In the case of an experiment, it could for example, facilitate the creation of a report by the simple selection and copying of the relevant embedded objects once the experiment has terminated. The organization of the Outcome AD reproduces the structure of the Description AD in terms of the structure of activities and tasks, but differs from it in the sense of having an `outcome` tag for each of the performed tasks. Furthermore, there is an outcome AD XML structure for each actor involved in the activities described above.

4 Using and Configuring the AD system

The details presented in this section currently present the way the AD system would be used independently of the LMS system included in DiViLab. This affects the way that a community AD would be specified (as detailed in section 4.1 *Working in the system as a learning scenario designer*), and will be updated in deliverable D7.7.

Run Tomcat and the database MySQL.

Open a navigator and connect to:

http://hostname:catalina_port/ADServerSetup

A window will show up that will allow us to create new ADs, as well as work with those already existing (figure 14).

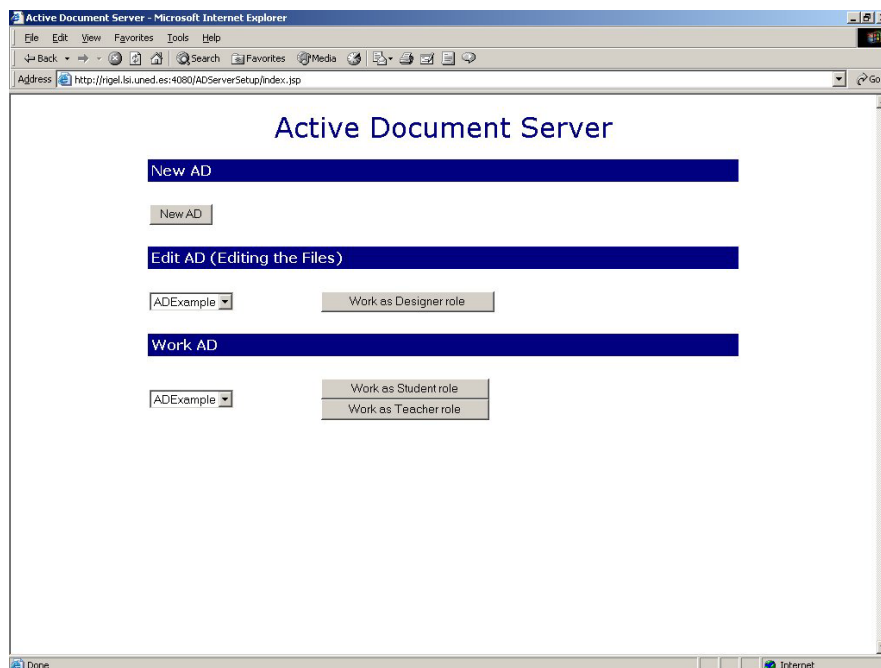


Figure 14. Accessing the ADServer

A user can interact with the The Active Document Server in three ways, as a scenario designer, as a student who is going to undertake the experiment, or as a teacher who is going to vigilate the students progress (using the Monitor) or correct the produced work. In the following three sections these three views of the system will be presented.

4.1 Working in the system as a learning scenario designer

After creating a new AD we can see that the structure is practically empty, it is only a template that we must modify to adapt it to our needs and to be able to obtain the AD that we really need.

To edit the AD that we have just created, we open a navigator and connect to: http://localhost:catalina_port/ADServerSetup

In the section “Edit AD (Editing the Files)” we select the AD that we want to edit and click on the button “Work as designer”, and then the “MetadataEditor” is opened and it allows us to edit the files that make up the selected AD (figure 15).

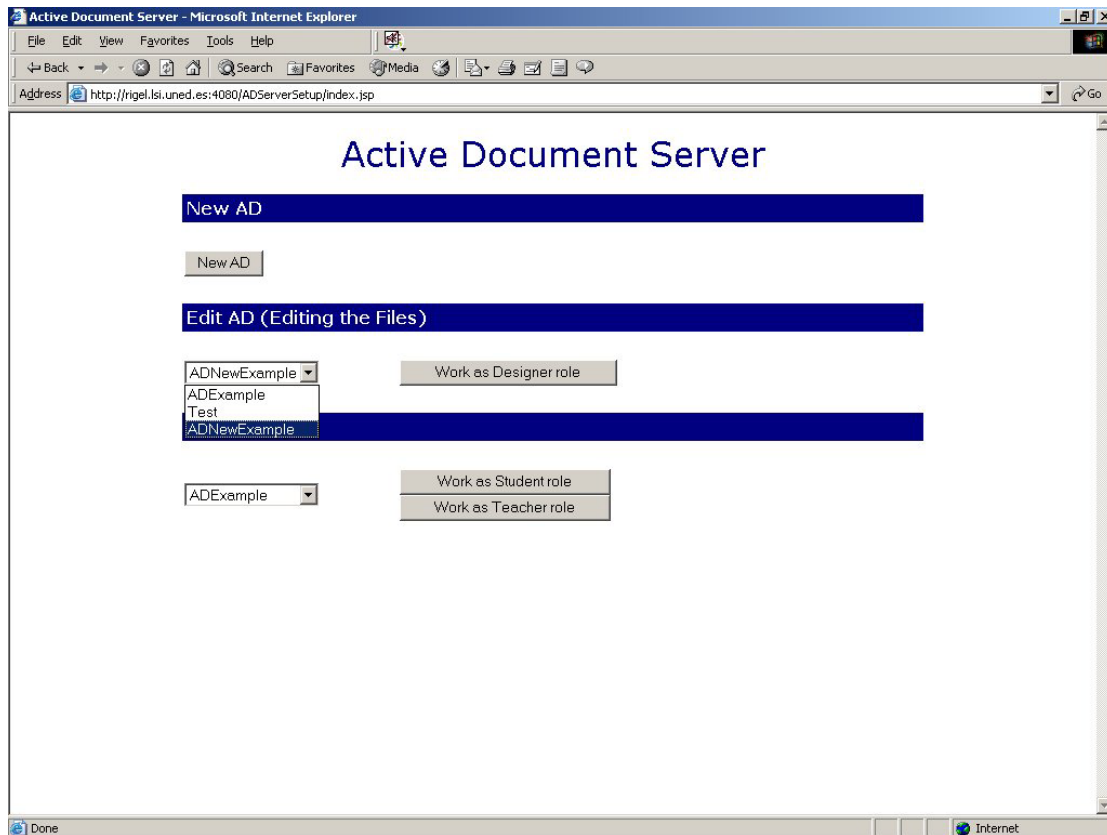


Figure 15. Choosing the AD to edit

To create and modify the prior XML documents, a metadata editor is provided. It assists in the process of defining learning environment and its associated resources in terms of ADs. Particularly definitions of activities, experiments, task, roles, and assignments of responsibilities to roles can be undertaken with this tool. This editor is divided into two frames:

The left frame offers a tree representation of the hierarchical structure of the XML document being edited. The frame on the right shows the node selected in said tree where 6 elements appear.

- The tool bar. This bar allows modifications on the structure of the XML document.
- The table of attributes. Through this table you can access the attributes of the node in course to edit and modify the value.
- The table of contents (entitled “Content Information”). The table of contents reference to all child nodes of the node that is being edited. This is how to navigate through the alternative XML document the tree hierarchy.
- The table of sibling nodes (entitled “See also”). This table shows the relationship of all the sibling nodes of the selected node. As before, its mission is to offer an alternative to navigation.

- The area of content editing. This area of text permits editing the content of the node that is terminal.

Out of the prior elements, it is worth highlighting the operations that the toolbar allows you to carry out to change the hierarchical structure of the document. These operations affect the nodes of the XML document. More specifically, it can:

- Create a child node
- Create a sibling node
- Clone a sub tree from a node
- Change the name of a node
- Erase a node

When selecting a node attribute, the frame on the right offers an area of text where you can edit and modify the content of the attribute as well as a toolbar with two possible actions referring to said node:

- Create a new attribute
- Erase the attribute that is being edited

With these elements, the editor can offer the designer the possibility to create and modify any structure of documents desired. But in order for the changes to be effective, the changes that have taken place must be saved. In order to do so, the upper frame of the editor has a button “Save XML” which allows you to save the changes remotely.

This editor does not check if the edited documents fulfil the DTD, it is very important for all files that compose the AD to be validated so they fulfil the DTDs from which they are based. If not, the AD will not work.

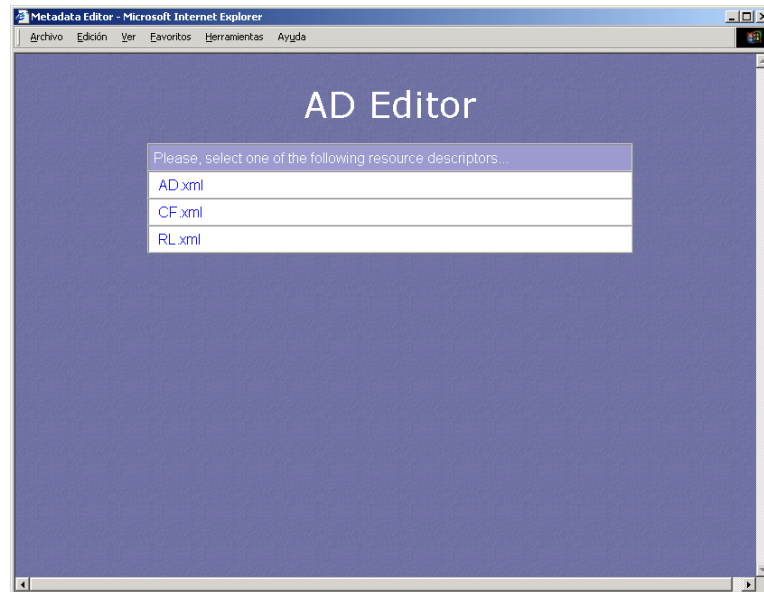


Figure 16. Selecting the AD File

When entering the design mode in the education environment, the web editor opens. As you can see in figure 16, first you can choose the document of configuration that you desire to edit. In this case, we select the entrance AD.xml.

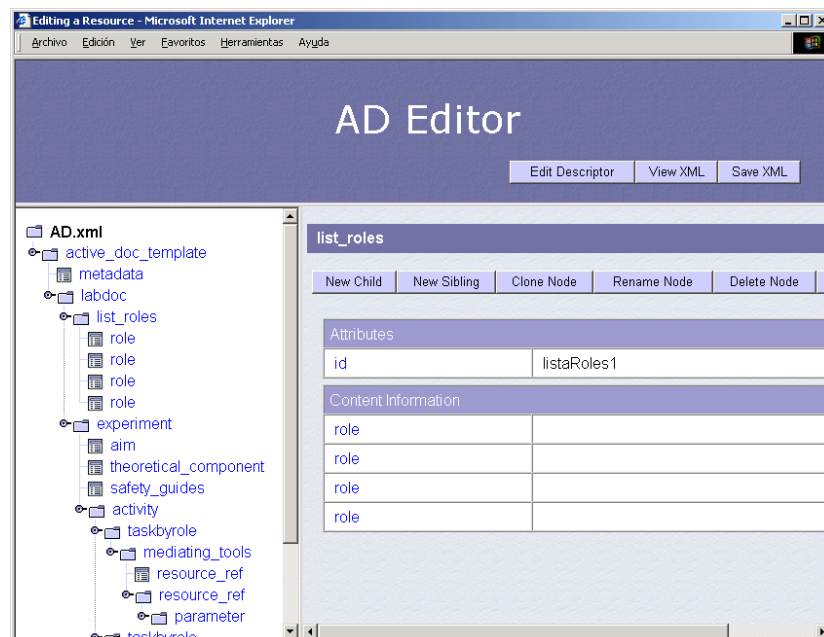


Figure 17. Adding roles to a new role list (*list-roles*)

Once the hierarchical tree corresponding to the AD.XML document is scrolled out and one of its nodes has been selected, the editor looks similar to figure 17. In this specific case, the node `list_nodes` is being edited. As can be seen in the toolbar, we are offered

the possibility to create new child and sibling nodes, erase the current node (`list_nodes`), rename it, or add a new attribute to it. The editor also offers the relationship of the node attributes (in this case only ID) and the relationship of child nodes (4 nodes role). If we select the attribute ID, a screen shows up where we can edit the value of said attribute.

In this screen we can see two buttons in the upper right corner of the right hand frame. One of them allows us to create a new attribute. The other one erases the attribute that is being edited. There is also an area of text where you can update the value of the attribute just by clicking on the button “Update”.

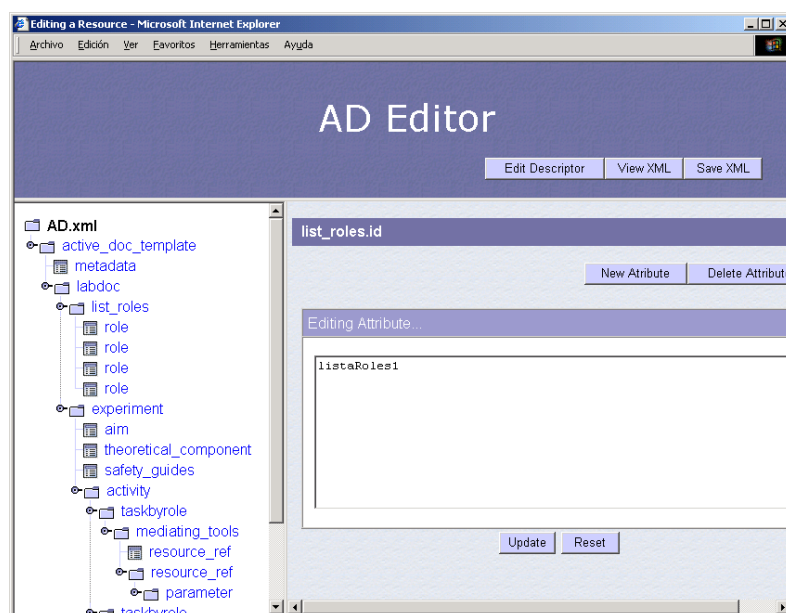


Figure 18. Editing the role list identifier (*id*)

Also, when we select one of the role nodes (figure 18) we go on to edit that node as shown in figure 19. As you can see in this case it is also possible to edit the content of the node as long as it is a leaf element within the hierarchy of the document.

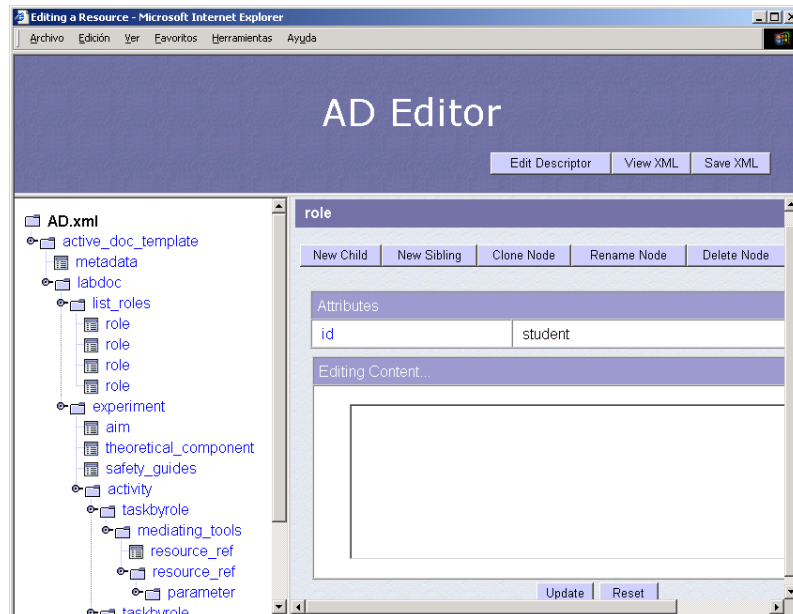


Figure 19. Editing a new role (student)

Before finishing the editing of the document, you must save the changes made. In order to do so, click on the button “Save XML” which can be found in the upper left side of the screen. This makes a window come up where all you need to do is click on the button OK.

4.1.1 The XML files which are used to define a scenario with the AD

An active document is made up of three XML files and one part of configuration:

- The actual active document
- The resource file
- The communities file
- The part of configuration that is stored in the XML file that contains the configuration of the ADServer.

The part of configuration and the three XML files are generated in the moment that the AD is created.

The three XML files that make up the AD are generated almost empty when a new AD is created, so the user must personalize them in order to adapt them to the desired AD.

These four files are related amongst themselves in the following manner:

- The AD uses references to resources that are found defined in the RL
- The RL defines tools that, in some cases, need variables from the AD, and in that case they can also be found defined in the configuration file.

- The CF makes references to experiments, activities, and tasks defined within the AD.

4.1.1.1 Tool types

A special distinction needs to be made about the types of tools that can be used in an AD, and as such we can distinguish between:

- General purpose
- Associated to tasks

These can also be:

- Local
- Remote

Furthermore, the above distinctions can be seen to be either internal (which refer to tools that can only be used with the AD System) or external (standalone applications that have been configured to work with the AD System). There is a series of local tools that are from the ADServer, that are defined within the XML files that make up the AD at the time of its creation and can be used there if desired. These are:

- **editor**: It allows you to carry out tasks that have written text answers. When you click on the button “Save”, it saves the answer in the results DB.
- **tooltest**: It allows you to carry out, answer and correct tasks that have multiple-choice type answers. When you click on the button “Save”, it stores the answer in the results DB.
- **DrawTool**: It allows you to carry out tasks that have drawn answers. When you click on the button “Save”, it stores the answer in the results DB.
- **correction**: It is a correction tool that allows the tutor to comment and qualify a task carried out by a student. When you click on “Save”, it stores the correction in the results DB.

4.1.2 The DescriptionAD

4.1.2.1 AD elements

The DescriptionAD is made up of the following elements:

Active_doc_template	
Element	<active_doc_template....>
Explanation	Element: An Active Document is composed of a variety of Laboratory Documents. Actually, a labdoc is an active document itself, but it can be considered to have a variety of labs to be carried out over the same active document. An active_doc_template bears also some metadata concerning the

	AD.
XML description	<!ELEMENT active_doc_template (metadata, labdoc)>
Example	<pre> <active_doc_template> <metadata>This is the metainformation</metadata> <labdoc id="labdoc01"> ... stuff cut for saving space see the following elements for details ... </labdoc> </active_doc_template> </pre>
Activity	
Element	<activity....>
Explanation	<p>Element: Performing an experiment implies carrying out a number of activities</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ id: the unique identifier of this activity ○ name: the activity's name ○ prerequisite: a reference to a prerequisite (a task which has to be carried out before starting this activity)
XML description	<pre> <!ELEMENT activity (#PCDATA resource_ref content_ref taskbyrole formatted_content tip object)*> <!ATTLIST activity id ID #REQUIRED name CDATA #REQUIRED prerequisite NMTOKEN #IMPLIED> </pre>
Example	<pre> <activity id="activity_002" name="Activity 2" prerequisite="pre2"> Description for Activity 2 ... stuff cut for saving space see the following elements for details ... </activity> </pre>
Editor window	See figures 29 to 32
Aim	
Element	<aim....>
Explanation	<p>Element: the Aim defines the objective of the experiment</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ label: the aim's identifier
XML description	<pre> <!ELEMENT aim (#PCDATA resource_ref content_ref object formatted_content tip)*> <!ATTLIST aim label CDATA #IMPLIED> </pre>
Example	<aim label="Aim"> Aim </aim>
Editor window	See figure 24
Bd_object	
Element	<bd_object....>

Explanation	<p>Element: a reference to a Data Base object</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ id: the unique identifier of this bd_object (reference) ○ domain: the object's category (as an object) ○ name: the referred object's identifier ○ category: the referred object's category ○ atr-label: the referred object's name ○ atr-content: the object's content ○ track: track the element's usage. No teacking by default ○ faq: include this element in a FAQ. No inclusion by default.
XML description	<pre><!ELEMENT bd_object (#PCDATA)> <!ATTLIST bd_object id ID #REQUIRED domain NMTOKEN #REQUIRED name NMTOKEN #REQUIRED category NMTOKEN #REQUIRED atr-label NMTOKEN #REQUIRED atr-content NMTOKEN #REQUIRED track (yes no) #IMPLIED faq (yes no) #IMPLIED></pre>
Example	<pre><bd_object atr-content="Content" atr-label="Name" category="Concept" domain="conceptual" id="obj4054" name="d031">tubo capilar </bd_object></pre>
Bd_relation	
Element	<bd_relation....>
Explanation	<p>Element: A reference to a DB relation</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ id: the unique identifier of this bd_relation reference ○ domain: the referred relation's domain ○ antecedent: the referred relation's antecedent ○ track: tracking (yes ot no). By default, there is no tracking ○ attrib: relation attribute ○ category: a domain category
XML description	<pre><!ELEMENT bd_relation (#PCDATA)> <!ATTLIST bd_relation id ID #REQUIRED name NMTOKEN #REQUIRED domain NMTOKEN #REQUIRED antecedent NMTOKEN #REQUIRED track (yes no) #IMPLIED attrib CDATA #IMPLIED category NMTOKEN #IMPLIED></pre>
Example	<pre><bd_relation id="rel200" name="belongs_to" domain="Conceptual" antecedent="gf_04" category="Compound">halógenos </bd_relation></pre>
Content_ref	

Element	<content_ref....>
Explanation	Element: Content_ref is a reference to an external element. This kind of content must be obtained as a result of a binding process or a DB access
XML description	<!ELEMENT content_ref (bd_object bd_relation glosary)>
Example	<pre> <content_ref> <glosary atr-content="Content" atr-label="Name" category="Concept" domain="Conceptual" id="glosary000284" order="alphabet">Glosario de Conceptos</glosary> </content_ref> </pre>
Experiment	
Element	<experiment....>
Explanation	<p>Element: Experiments are composed by:</p> <ul style="list-style-type: none"> - The list of resources of the learning environment -A structure model to include content and activities <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - name: the name of this activity, an identifier? - title: - prerequisite: An experiment may have as a prerequisite another experiment -prerequisite_type: either ‘delivered’, meaning that the experiment has been given to the students? Or ‘assessed’, meaning that it’s been evaluated
XML description	<pre> <!ELEMENT experiment (aim?, theoretical_component?, safety_guides?, activity+)> <!ATTLIST experiment name ID #REQUIRED title CDATA #REQUIRED prerequisite IDREF #IMPLIED prerequisite type (assessed delivered) #IMPLIED> </pre>
Example	<pre> <experiment name="exp_1" title="Experiment 1" prerequisite="pre1"> <aim label="Aim"> Aim </aim> <theoretical_component label="Theorical Component"> Theorical Component </theoretical_component> <safety_guides label="Safety Guides"> Safety Guides </safety_guides> <activity id="activity_001" name="Activity 1" prerequisite="pre1"> ... stuff cut for saving space see the following elements for details ... </activity> </experiment> </pre>
Editor window	See figures 21 to 28
Glossary	

Element	<glossary....>
Explanation	<p>Element: An external glossary defining terms and concepts from the experiment's domain</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the unique identifier of this glossary's element o category: the glossary's element's category (within the glossary's domain) o domain: the kind of object which this glossary entry is defining o atr-label: the glossary's element name o atr-content: the glossary's element content o track: whether to track the use of this element. No tracking by default o order: the terms sorting order, either alphabetical or in the input sequence
XML description	<pre><!ELEMENT glossary (#PCDATA reference)*> <!ATTLIST glossary id ID #REQUIRED category NMTOKEN #REQUIRED domain NMTOKEN #REQUIRED atr-label NMTOKEN #REQUIRED atr-content NMTOKEN #REQUIRED track (yes no) #IMPLIED order (alphabet sequence) #IMPLIED></pre>
Example	<pre><glossary atr-content="Name" atr-label="Name" category="FunctionalGroup" domain="Conceptual" id="glosary008029" order="alphabet">glosarios por Grupo Funcional <reference atr-content="Name" category="Compound" domain="Conceptual" id="glosary080030" name="belongs_to" position="consequent">Nombre de los compuestos pertenecientes a este grupo funcional </reference> <reference atr-content="Content" category="TableIR" domain="Conceptual" id="glosary008031" name="belongs_to" position="antecedent">Frecuencias de absorción de los compuestos pertenecientes a este grupo funcional </reference> </glossary></pre>
Labdoc	
Element	<labdoc....>
Explanation	<p>Element: A laboratory document defines one or more experiments.</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - id: the document unique identifier
XML description	<pre><!ELEMENT labdoc (list_roles*, list_prerrequisites*, experiment+)> <!ATTLIST labdoc</pre>

	id ID #REQUIRED>
Example	<pre> <labdoc id="labdoc01"> <list_roles id="RoleList1"> ... stuff cut for saving space see the following elements for details ... </list_roles> <experiment name="exp_1" title="Experiment 1"> ... stuff cut for saving space see the following elements for details ... </experiment> </labdoc> </pre>
List_prerequisites	
Element	<list_prerequisites...>
Explanation	<p>Element: list_prerequisites is the set of conditions that must be held before a task can be started.</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - id: the list identifier
XML description	<pre> <!ELEMENT list_prerequisites (prerequisite+)> <!ATTLIST list_prerequisites id ID #REQUIRED> </pre>
Example	<pre> <list_prerequisites id="listaPre1"> <prerequisite id="pre1" date_in="2003-6-16" date_out="2003-6-23"/> <prerequisite id="pre2" ref="activity_001" type="passed"/> </list_prerequisites> </pre>
Editor window	See figures 37 to 39
List_roles	
Element	<list_roles....>
Explanation	<p>Element: the set of roles taking part in the experiments within that particular labdoc. Later on, these roles will be bound to taskbyrole elements.</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the list identifier
XML description	<pre> <!ELEMENT list_roles (role+)> <!ATTLIST list_roles id ID #REQUIRED> </pre>
Example	<pre> <list_roles id="RoleList1"> <role id="student"/> <role id="teacher"/> <role id="guest"/> <role id="marker"/> </list_roles> </pre>
Editor window	See figures 17 and 18
Mediating_tools	
Element	<mediating_tools....>
Explanation	Element: list of references to the helping tools used for the different tasks

XML description	<!ELEMENT mediating_tools (resource_ref+)>
Example	<pre> <mediating_tools> <resource_ref display="insite" height="500" id="ref666" id_ref="correction_tool" label="" width="300"> <parameter> <param name="just" value="Right#Average#Wrong"/> <param name="values" value="10#5#0"/> </parameter> </resource_ref> </mediating_tools> </pre>
Editor window	See figure 46
Metadata	
Element	<metadata....>
Explanation	Element: Representation of Metadata can use XML LOM binding from IEEE LTSC or any other. Also DC has an XML binding
XML description	<!ELEMENT metadata (#PCDATA)>
Example	<metadata>This is the metainformation</metadata>
Object	
Element	<object....>
Explanation	<p>Element: Definition of the generic content:</p> <ul style="list-style-type: none"> - Objects - References to objects or content <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the identifier of the object?? o label: to show whenever the user moves the pointer over the object's link or when it is displayed on another window, as the window's title o format: in which the object has been typeset o display: whether it is to be displayed in the current or another window
XML description	<pre> <!ELEMENT object (table map graphic)> <!ATTLIST object id ID #IMPLIED label CDATA #REQUIRED format (latex HTML smiles) #IMPLIED display (insite otherWindow) #REQUIRED> </pre>
Example	<pre> <object display="insite" label="Solubilidad"> <graphic height="150" uri="http://rigel.lsi.uned.es:4080/ad1_des/ jsp/parser_jsp/images/ADQuimica/solubilidad.jpg" width="400"/> </object> </pre>
Param	
Element	<param....>

Explanation	<p>Element: A param is an attribute-value pair which can be passed as a parameter to a resource which needs it</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o name: the attribute's name o value: the optional value for initialising the attribute
XML description	<pre><!ATTLIST param name CDATA #REQUIRED value CDATA #IMPLIED></pre>
Example	<pre><param name="text" value="Comments"/></pre>
Editor window	<i>See figures 53 and 54</i>
Parameter	
Element	<pre><parameter....></pre>
Explanation	Element: It's a list (could be empty) of params (attribute-value pairs)
XML description	<pre><!ELEMENT parameter (param*)></pre>
Example	<pre><parameter> <param name="text" value="Comments"/> </parameter></pre>
Editor window	<i>See figures 51 and 52</i>
Prerequisite	
Element	<pre><prerequisite....></pre>
Explanation	<p>Element: a prerequisite is a condition which must be held before a task can start</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the prerequisite identifier o ref: a reference to an experiment or activity o type: the task's status, either 'correct', 'done' or 'passed' o date_in: starting date o date_out: finishing date
XML description	<pre><!ELEMENT prererequisite EMPTY> <!ATTLIST prererequisite id ID #REQUIRED ref IDREF #IMPLIED type (correct done passed) #IMPLIED date_in CDATA #IMPLIED date_out CDATA #IMPLIED></pre>
Example	<pre><prerequisite id="pre1" date_in="2003-6-16" date_out="2003-6-23"/></pre>
Editor window	<i>See figures 40 to 45</i>
Reference	
Element	<pre><reference....></pre>
Explanation	<p>Element: a reference to an external element, a relation to an external source</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the unique identifier of this reference

	<ul style="list-style-type: none"> ○ name: the reference's name as a relation ○ domain: the domain to which the relation is established ○ category: a category within the previous domain ○ atr_content: the reference content's name ○ position: either antecedent or consequent
XML description	<pre><!ELEMENT reference (#PCDATA)> <!ATTLIST reference id ID #IMPLIED name NMTOKEN #REQUIRED domain NMTOKEN #REQUIRED category NMTOKEN #IMPLIED atr-content NMTOKEN #REQUIRED position (antecedent consequent) #IMPLIED></pre>
Example	<pre><reference atr-content="Content" category="TableIR" domain="Conceptual" id="glosary008031" name="belongs_to" position="antecedent">Frecuencias de absorción de los compuestos pertenecientes a este grupo funcional </reference></pre>
Resource_ref	
Element	<resource_ref....>
Explanation	<p>Element: A reference to a resource defined in the resources file (resources.xml)</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ id: the unique identifier of this resource_ref ○ Id_ref: the referred resource's name ○ label: a text to display when the resource is used ○ width: the resource is to be displayed in a rectangle this width ○ height: the resource is to be displayed in a rectangle this height ○ display: whether it is to be displayed in the current or another window
XML description	<pre><!ELEMENT resource_ref (parameter?)> <!ATTLIST resource_ref id ID #REQUIRED id_ref CDATA #REQUIRED label CDATA #IMPLIED width CDATA #IMPLIED height CDATA #IMPLIED display (insite otherWindow) #REQUIRED></pre>
Example	<pre><resource_ref display="insite" height="500" id="ref667" id_ref="correction_tool" label="Elige la opcion" width="300"> <parameter> <param name="just" value="Right#Average#Wrong"/> <param name="values" value="10#5#0"/> </parameter> </resource_ref></pre>

Editor window	<i>See figures 47 to 50</i>
Role	
Element	<role....>
Explanation	<p>Element: each of the parts that can be taken while using the labdoc, i.e., while carrying out experiments</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the role's identifier
XML description	<pre><!ELEMENT role EMPTY> <!ATTLIST role id ID #REQUIRED></pre>
Example	<pre><role id="student"/></pre>
Editor window	<i>See figure 19</i>
Safety_guides	
Element	<safety_guides....>
Explanation	<p>Element: it describes the safety rules or considerations to take into account while performing this experiment</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - label: the element's identifier
XML description	<pre><!ELEMENT safety_guides (#PCDATA resource_ref content_ref object formatted_content tip)*> <!ATTLIST safety_guides label CDATA #IMPLIED></pre>
Example	<pre><safety_guides label="Safety Guides"> Safety Guides </safety_guides></pre>
Editor window	<i>See figure 28</i>
Taskbyrole	
Element	<taskbyrole....>
Explanation	<p>Element: This element defines, for each task, the roles which can perform them</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the unique identifier of this taskbyrole o name: the task name o description: the task's description o prerequisite: a reference to the task prerequisite (another task which has to be done before this one) o roles: a set of roles which can be involved in this task o markerTask: a reference to the marking task for this one if it is needed
XML description	<pre><!ELEMENT taskbyrole (#PCDATA resource_ref content_ref mediating_tools formatted_content tip)*> <!ATTLIST taskbyrole id ID #REQUIRED</pre>

	<pre> name CDATA #REQUIRED description CDATA #IMPLIED prerequisite IDREFS #IMPLIED roles IDREFS #REQUIRED markerTask IDREF #IMPLIED </pre>
Example	<pre> <taskbyrole id="taskdef021" name="Task 1" roles="student teacher" markerTask="taskdef021_m"> What is a Flow Diagram? Draw one and comment it. <mediating_tools> <resource_ref id="EditorMAref1" id_ref="Drawtool" display="insite" label="Draw Editor" height="300" width="300"/> <resource_ref id="ref7" id_ref="FormularioEditor" display="insite" label="Text Editor"> <parameter> <param name="text" value="Comments"/> </parameter> </resource_ref> </mediating_tools> </taskbyrole> </pre>
Editor window	<i>See figures 33 to 36 as well as 55 to 57</i>
Theoretical_component	
Element	<theoretical_component....>
Explanation	<p>Element: the theoretical component is the necessary domain knowledge for the experiment, its background</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> - label: the element's identifier
XML description	<pre> <!ELEMENT theoretical_component (#PCDATA resource_ref content_ref object formatted_content tip)*> <!ATTLIST theoretical_component label CDATA #IMPLIED> <!-- Agosto 2001 a�adi tool_ref al theoretical_component. Beti ... No aparece aqu� ... --> </pre>
Example	<pre> <theoretical_component label="Theoretical Component"> Theoretical Component </theoretical_component> </pre>
Editor window	<i>See figure 28</i>
Tip	
Element	<tip....>
Explanation	<p>Element: A tip is a link in the AD that leads the user to an explanation more in depth of a detail</p>
XML description	<pre> <!ELEMENT tip (#PCDATA tip_content)*> </pre>
Example	<pre> <tip>velocidad de disoluci�n <tip_content id="q123">Es la velocidad a la que la sustancia se disuelve </tip_content> </tip> </pre>
Tip_content	

Element	<tip_content....>
Explanation	<p>Element: The actual content of a tip, either a text, a content_ref, an object or a formatted_content</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the unique identifier of this content
XML description	<pre><!ELEMENT tip_content (#PCDATA content_ref object formatted_content)*> <!ATTLIST tip_content id ID #REQUIRED></pre>
Example	<pre><tip_content id="tip00000012"> hidroxilo (-OH) carboxilo (-COOH) amino (-NH2) éter (-O-) </tip_content></pre>

General notice: hyphens are not admitted within the value of the identifiers

4.1.2.2 Editing the AD-definition XML file

When editing the file AD.xml using the MetadataEditor it is very important to fulfil the dtd called “AD_ActiveDoc_v7.dtd” because if it is not fulfilled, the AD will not work.

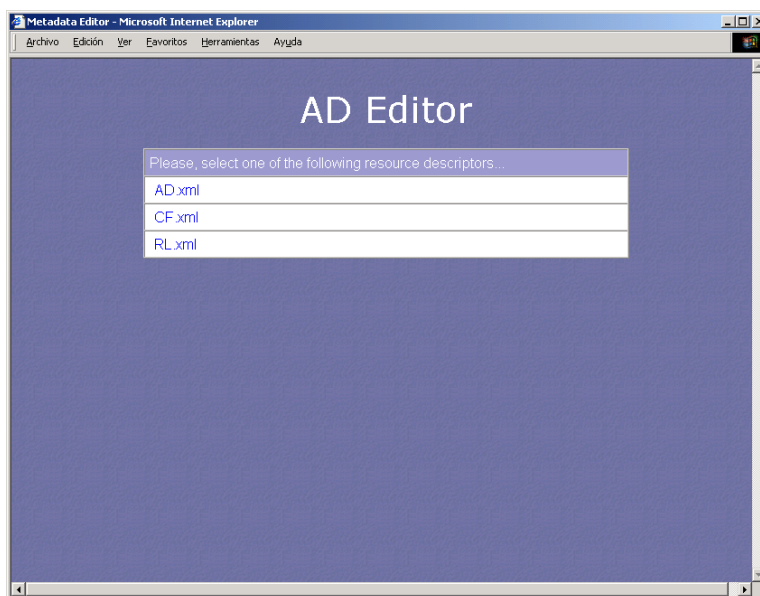


Figure 20. Editing the AD Files

When entering the design mode of the educational environment, the metadata editor described previously is opened. In the first place you can choose the configuration

document that you would like to edit. In this case we must select the first link corresponding to the document AD.xml that we want to edit (figure 20). Throughout the next lines we will describe the configuration actions that are most common and can be done on the document.

Inserting a new experiment

The first configuration task in the environment is the configuration of a new experiment. As shown in the active document that appears by defect already has an experiment (figure 21).

We must replicate the structure. In order to do so, the fastest option is to go over to the node “experiment” and click on the button “Clone Node”. As a result, a new experiment identical to the previous one is created. Then the nodes of the experiment can be edited and correctly configured.

However, we will detail step by step the creation of a new experiment. In order to do so, the following steps must be followed:

1. Go on the labdoc node

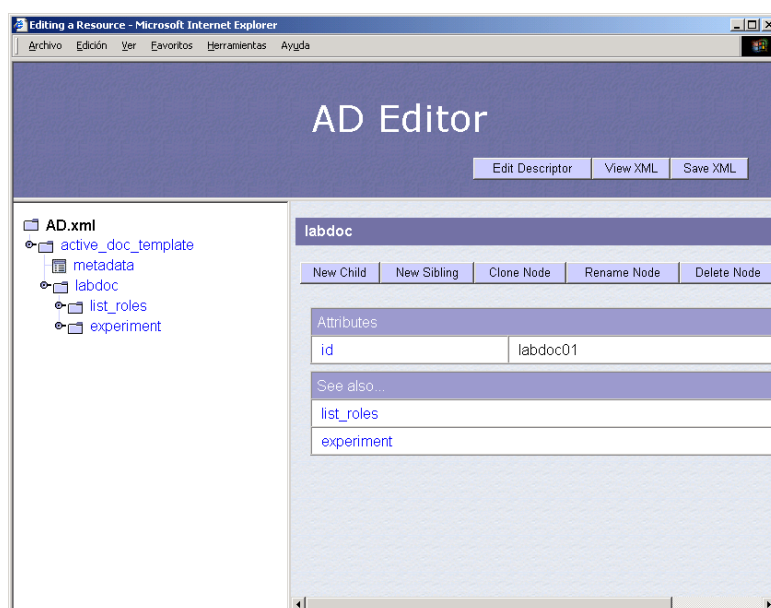


Figure 21. Selecting a *labdoc* node

2. Click the button New Child of the button on the right. A text square will show up asking for the name of the new node (figure 22). Type in the word “experiment”.

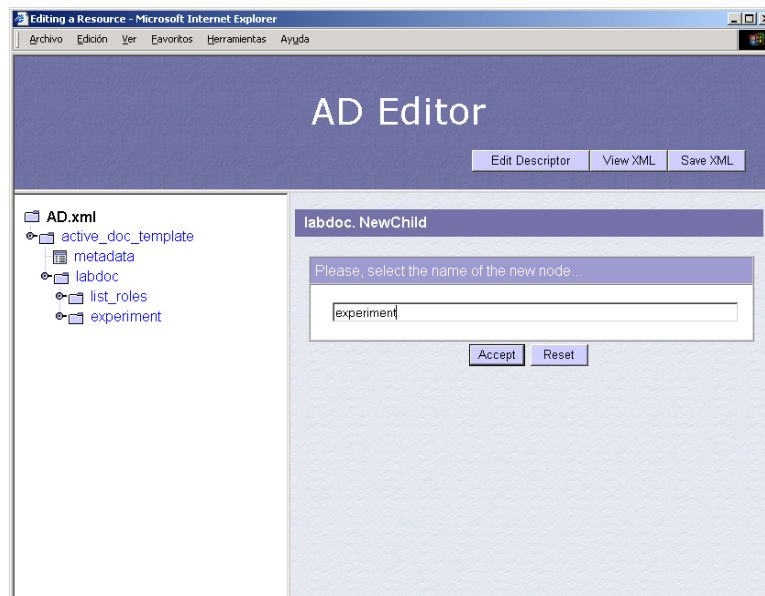


Figure 22. creating a new experiment node

3. Click on the button Accept. As a result, a new node terminal will be generated of the type “experiment”. Go to this new node and repeat step two to create three child nodes. These nodes correspond to the terminal elements “aim”, “theoretical_components” and “safety_guides”. As you can see, a sub structure similar to the proceeding experiment has been created (figure 23).

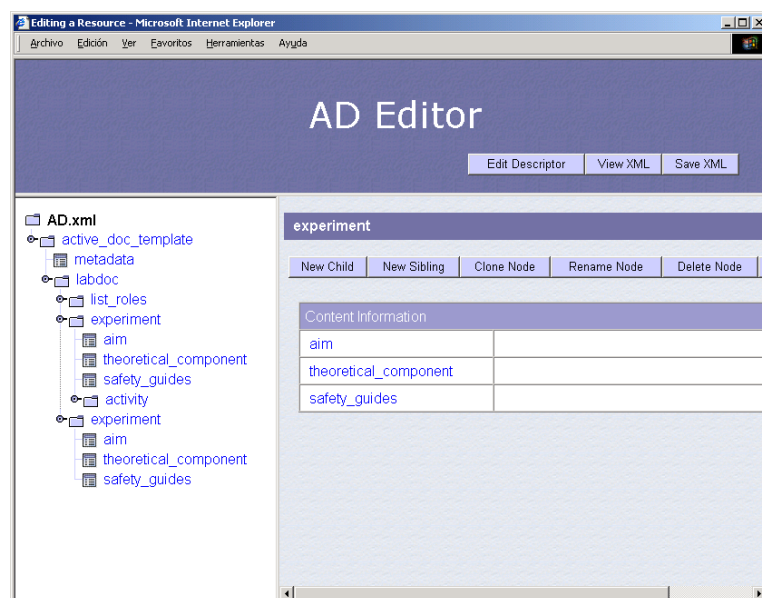


Figure 23. Structure for a new experiment

4. Now select each one of these child nodes and assign them a value that according to its meaning. For example, to specify the objectives of the experiment, click on “aim” in the

link in the table in the right hand frame or in the tree on the left hand side and specify a value in the text area (figure 24). To accept, click the button “Accept”.

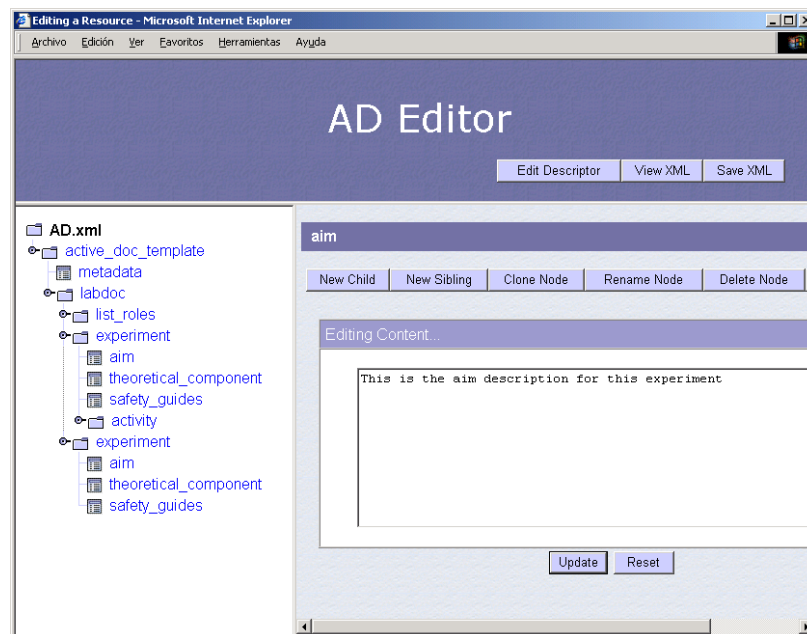


Figure 24. Filling in the content for the aim attribute

5. Select the other two nodes of the experiment and repeat the process described in step 4, when finished, we will have the one that appears in figure 25.

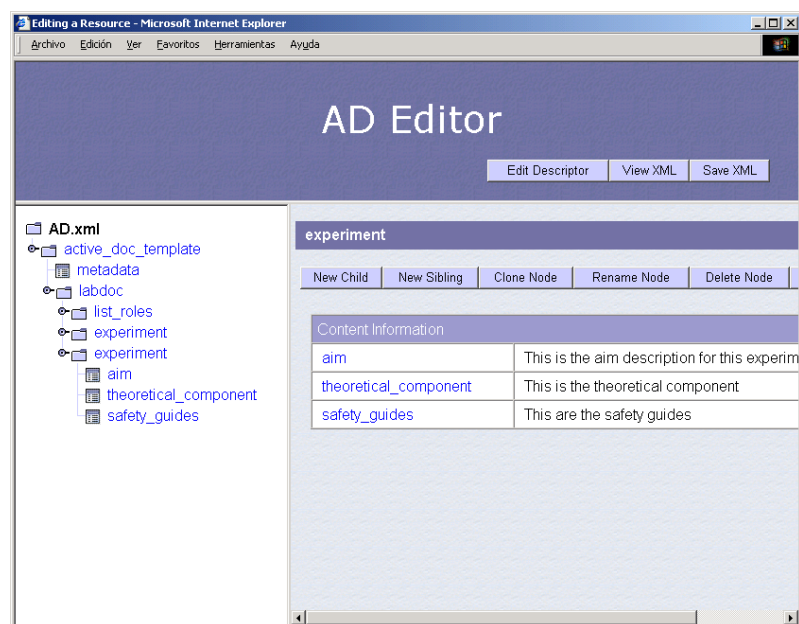


Figure 25. The content information attributes for an experiment

6. The description of an experiment also requires the addition of two attributes: the name (name) and the title. To enter them, go to the node that corresponds to the experiment and click on “New Attribute” (figure 26). Write the word “name” in the text square and click on the “Accept” button.

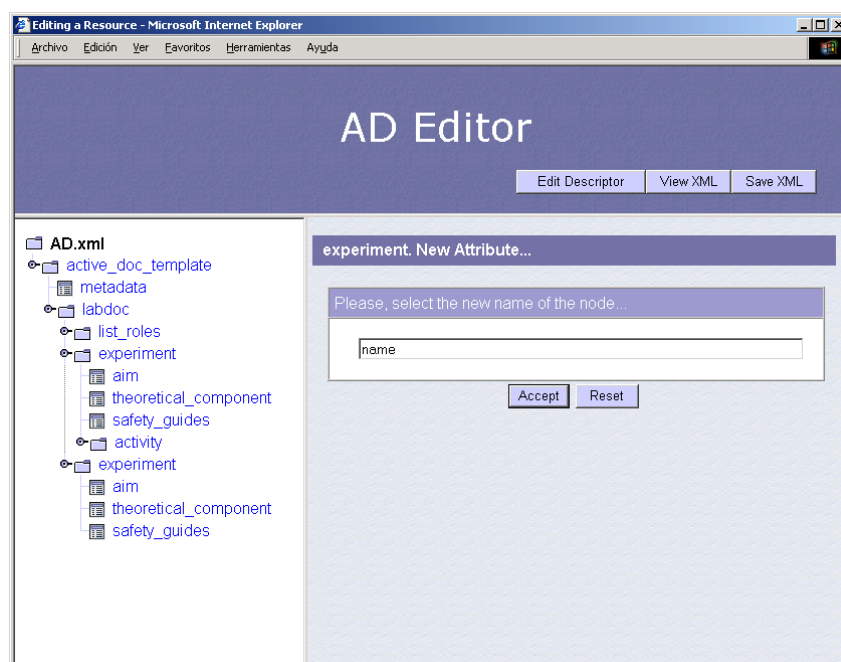


Figure 26. Adding the attribute *name* to a new experiment

7. Now repeat this process for the attribute “title”. As a result you will see that if the experiment is selected, there are two new attributes without value. To give them a value, follow each of the links in the table “Attributes” in the frame on the right. A window will appear with an area for text where you can assign value to the attribute (figure 27).

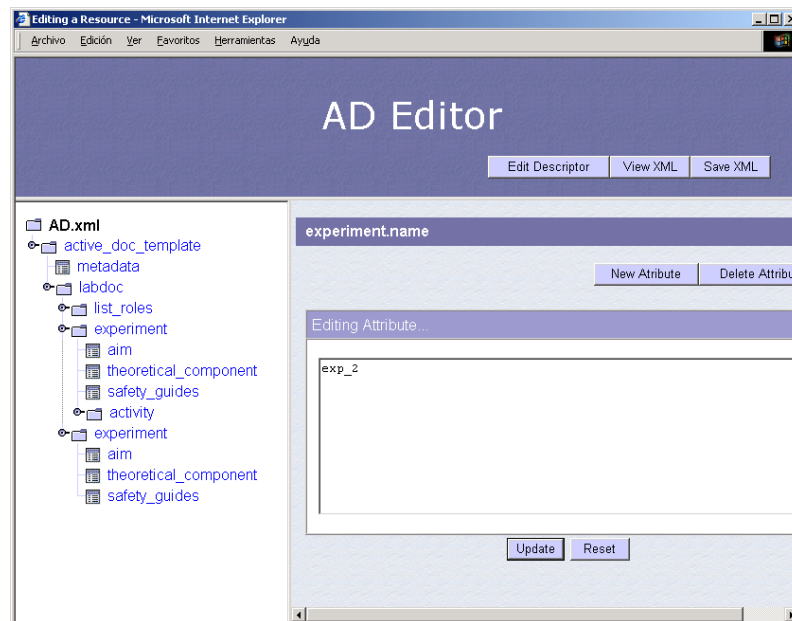


Figure 27. Giving the name to an experiment

8. Repeat the previous process for the other attribute. As a result, you should get a configuration for a new experiment similar to the one shown in figure 28.

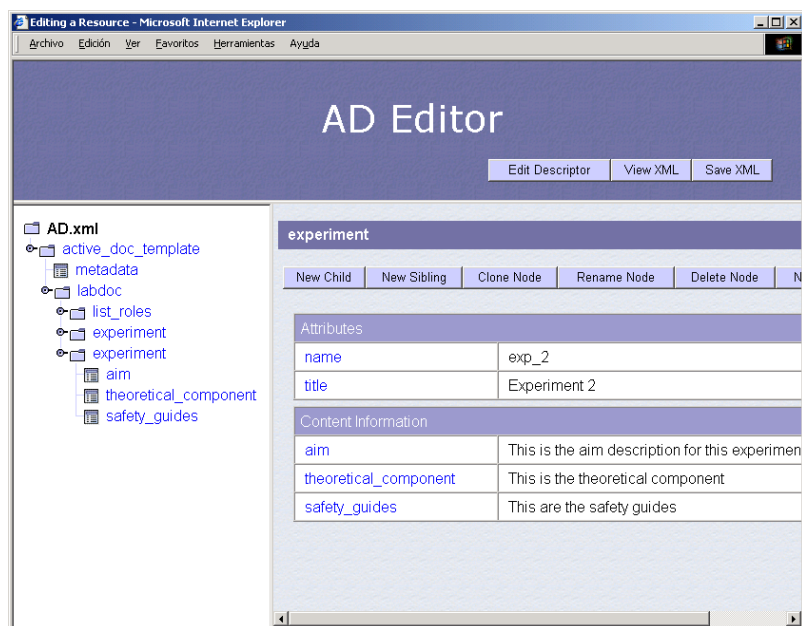


Figure 28. Experiment structure: attributes and content information

Entering a new activity

The following is a description of how an activity can be added to the previous experiment using the editor. Follow these steps:

1. Go to the experiment that you would like to incorporate a new activity to and click “New Child” on the frame on the right. On the right frame on the right type the word “activity” (figure 29) and click the button “Accept”.

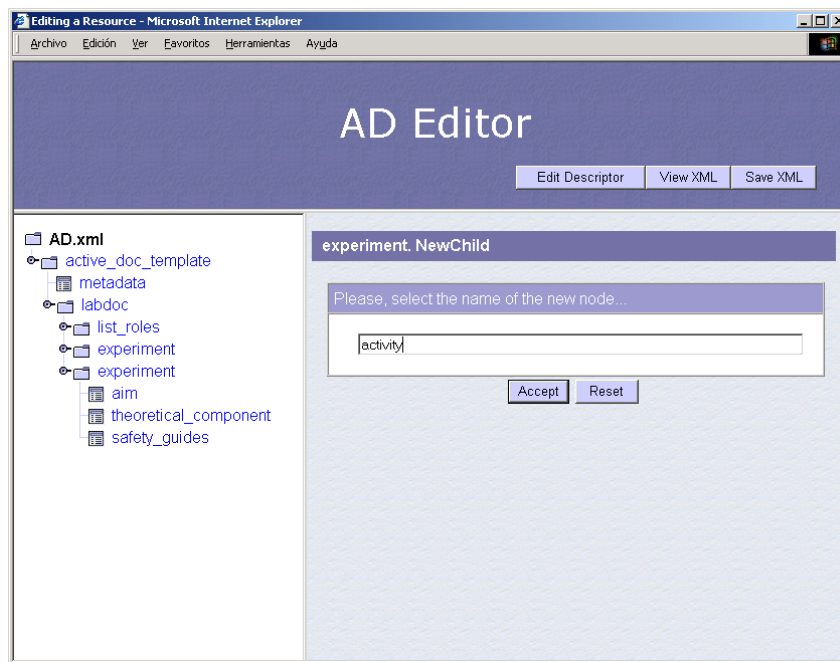


Figure 29. Adding a new activity to an experiment

2. Go to the node activity recently created. Add two attributes to the node: name and title. To do so, click the button “New Attribute”, type the word “name” (figure 30) and click on the button “Accept”. Repeat the process for the other attribute.

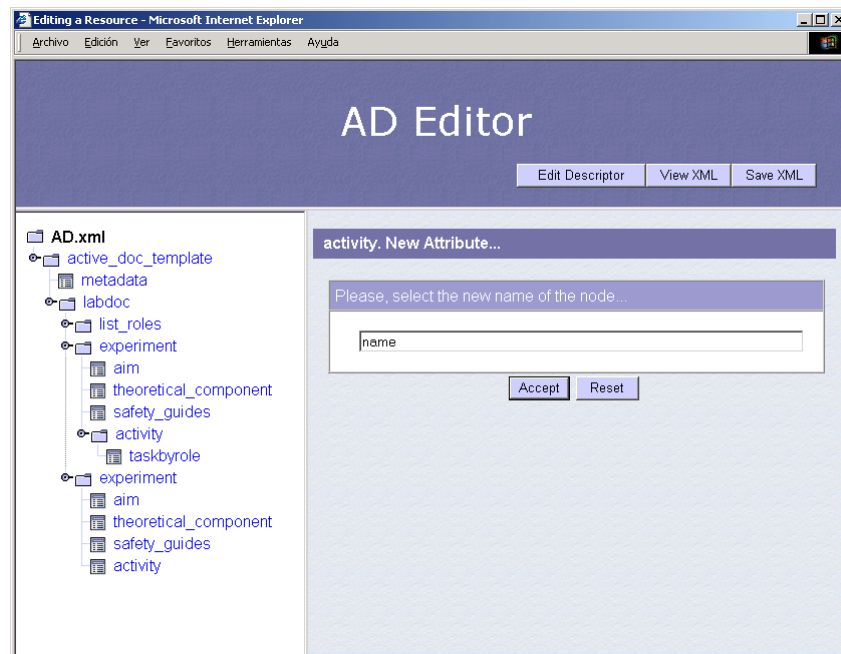


Figure 30. Creating the attribute *name* for a new activity

3. Go to the activity node and follow the link of each of the attributes on the table “Attributes” on the frame on the right to assign them a value. Type in a value in the text area (figure 31) and click on “Accept”. Repeat the process for the other attribute.

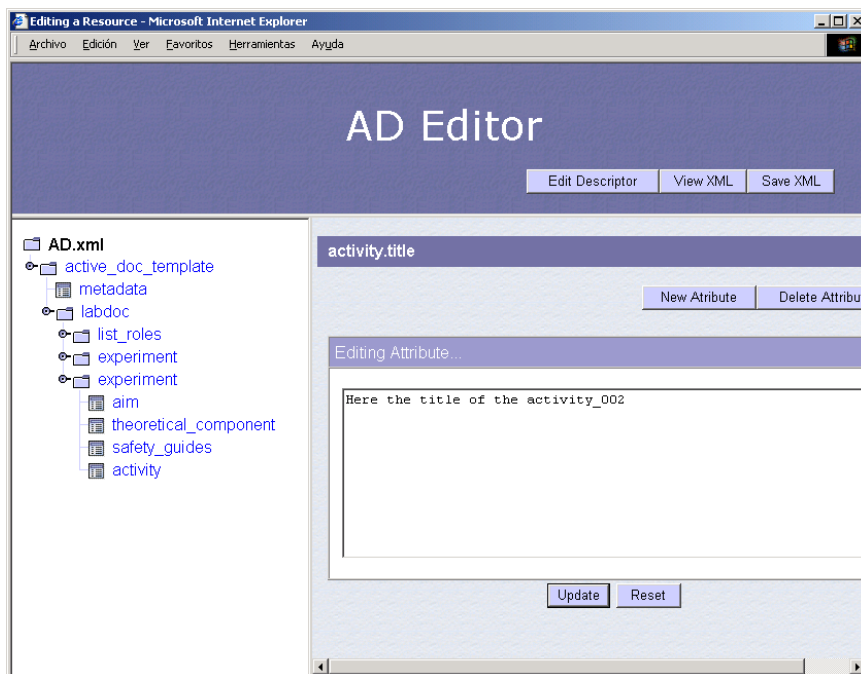


Figure 31. Filling in the title of a new activity

4. As a result, the newly created activity should look like the example shown in figure 32.

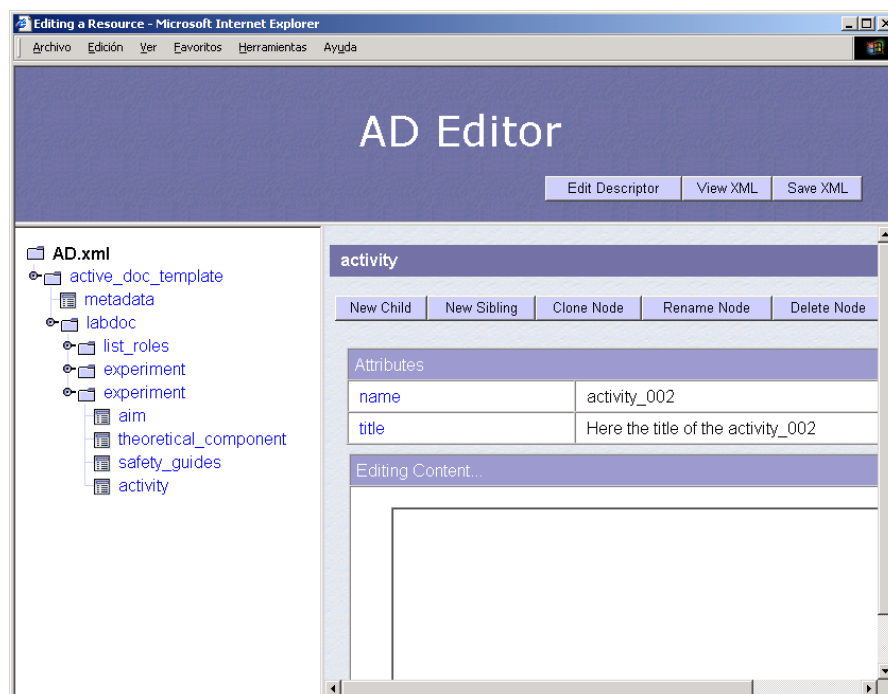


Figure 32. A new activity. Attributes *name* and *title* and their values

Entering a new task

In this section we describe the steps to enter a new task into an activity using the editor. For this, we create a new node with the name “taskbyrole” that depends on the activity that it belongs to. The process is summarized in the following steps:

1. Go to the node that corresponds to the activity that you would like to add the task to. Click on the button “New Child” and type in the word “taskbyrole” in the text square on the right (figure 33). Then click on the button “Accept”.

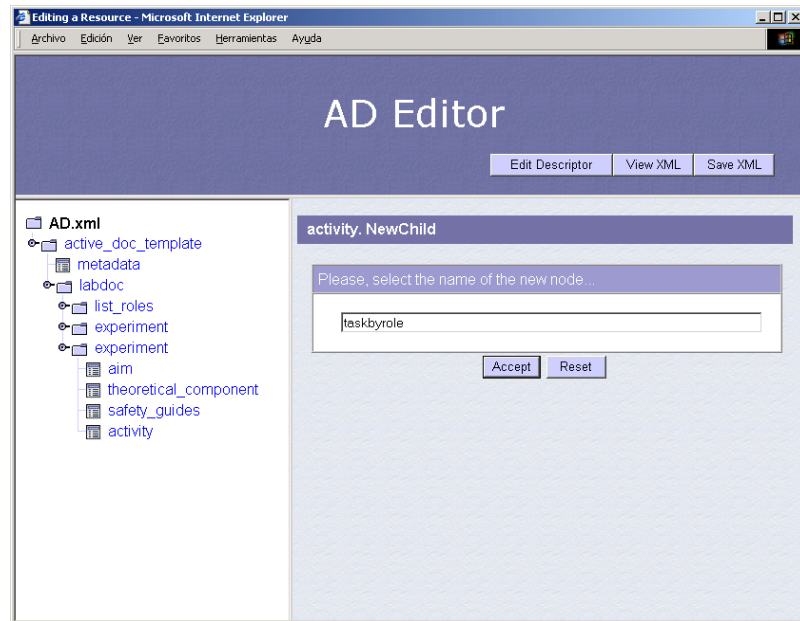


Figure 33. Creating a new *taskbyrole* node for an activity

2. Now go to the recently created node. Three new attributes must be created: an identifier (“id”), a name (“name”) and a list of roles (“roles”). Click on the button “New Attribute” and write in the text frame the word “id” (figure 34). Then click on Accept. Repeat this process for the other two attributes.

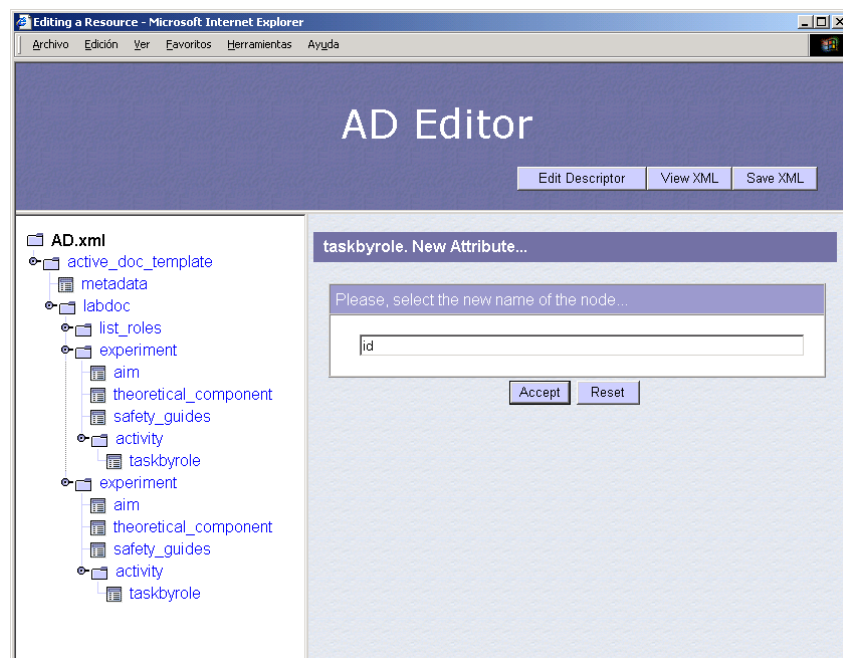


Figure 34. Adding an *identifier* to a new *taskbyrole*

3. Go back to the node created “taskbyrole” and select one of the links that appear in the table “Attributes” in the right frame. Type the value that you want to assign to the attribute in the text area that comes up (figure 35) and click on the Accept button. Repeat the process for the other two attributes. Notice that the attribute roles correspond to a collection of names. These must be typed in separately in blanks in the text frame and the possible values must be defined within list_roles. The following image illustrates this.

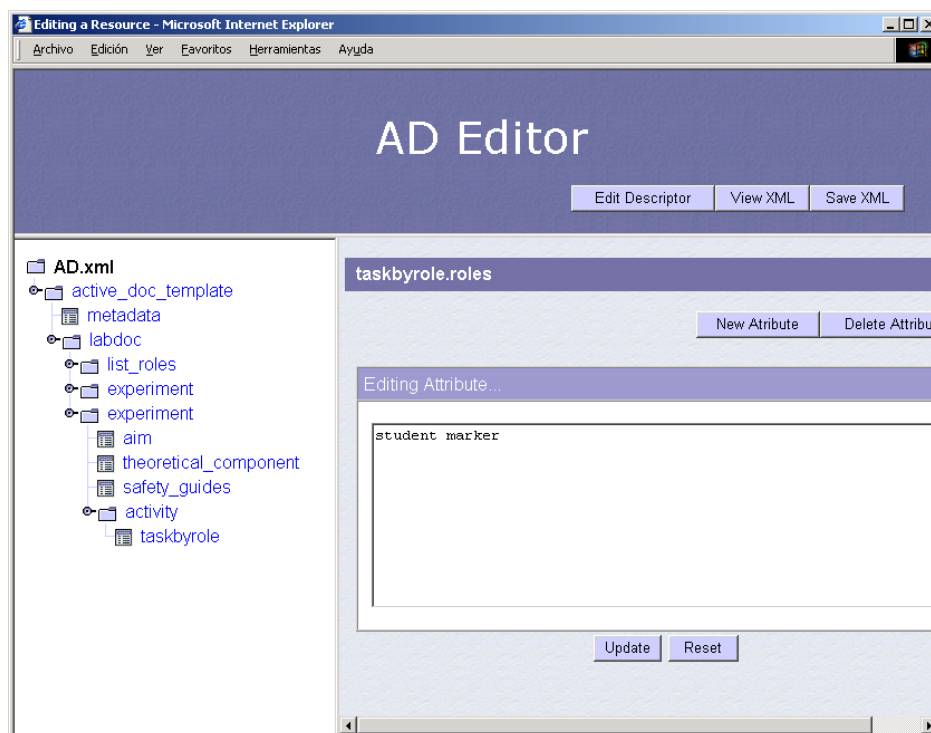


Figure 35. Adding a list of possible roles for a task to a *taskbyrole* element

4. Finally the task should be translated in the node taskbyrole similar to figure 36.

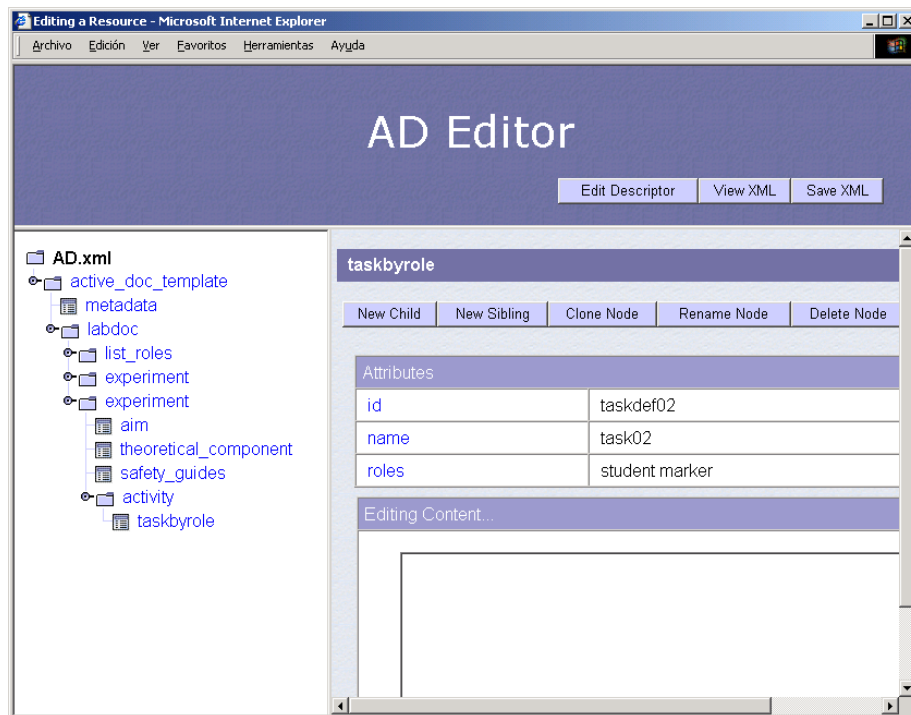


Figure 36. Inspecting the attributes of a new *taskbyrole* element

4.1.2.3 Definition of prerequisites

Another relevant aspect of building a scenario is the addition of prerequisites associated with tasks, activities and experiments. These prerequisites can be of different kinds. They can specify temporary starting and ending periods, or conditions regarding other experiments, activities or tasks.

The definition of a prerequisite is the following:

- Starting date: The date when access is permitted indicated by the value in the attribute “date_in” using the following format: year-month-day
- Ending date: Date when access is no longer permitted indicated by the value in the attribute “date_out” using the following format: year-month-day
- A specific part of the AD (experiment, activity or taskbyrole) be:
 - correct: The tutor has corrected that part of the AD
 - done: The student has carried out that part of the AD
 - passed: The qualification of that part is five or above

The desired value is indicated in the attribute “type” and the attribute “ref” indicates the id of part of the AD (experiment, activity or taskbyrole)

To define the prerequisites within the AD we have to carry out the following steps:

- Put all the definitions of prerequisites that you desire to use in the AD within the `<list_prerequisites>`.
- Then within the section we want attached to the prerequisite we use the attribute “prerequisite” and we give it the value “id_prerequisite” that refers to the prerequisite we want to apply to this part.

Example:

```
.
.
<list_prerequisites id="listaPre1">
    <prerequisite id="pre1" date_in="2003-3-4" date_out="2003-3-12"/>
    <prerequisite id="pre2" ref="exp_1" type="passed"/>
</list_prerequisites>
.
.
<experiment name="exp_1" title="Experiment 1" prerequisite="pre1">
.
.
<experiment name="exp_2" title="Experiment 2" prerequisite="pre2">
.
.
.
```

We will illustrate the addition of prerequisites adding the two following prerequisites.

- A date prerequisite
- A prerequisite that you must first have experiment exp1 finished in order to be able to carry out experiment exp2.

To carry this out, the first thing that we must do is declare the collection of prerequisites, and then associate them to the experiments, activities or tasks that we desire. The following are the steps to follow:

1. Go to the first experiment and click on the button “New Sibling”. Type in “list_prerequisites” (figure 37) in the text square that appears and click on the Accept button.

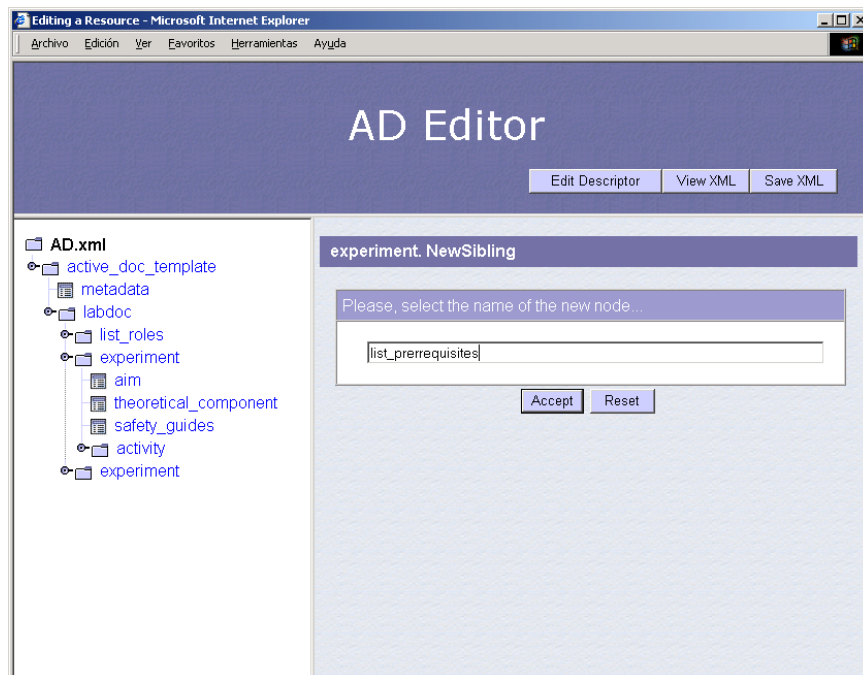


Figure 37. Adding a new list of prerequisites (*list_prerequisites*) to an experiment

2. Create a new name id attribute on the recently created node. To do so, con on the node and click on the button “New Attribute”. In the text square that appears, type in the word “id” (figure 38) and click on “Accept”.

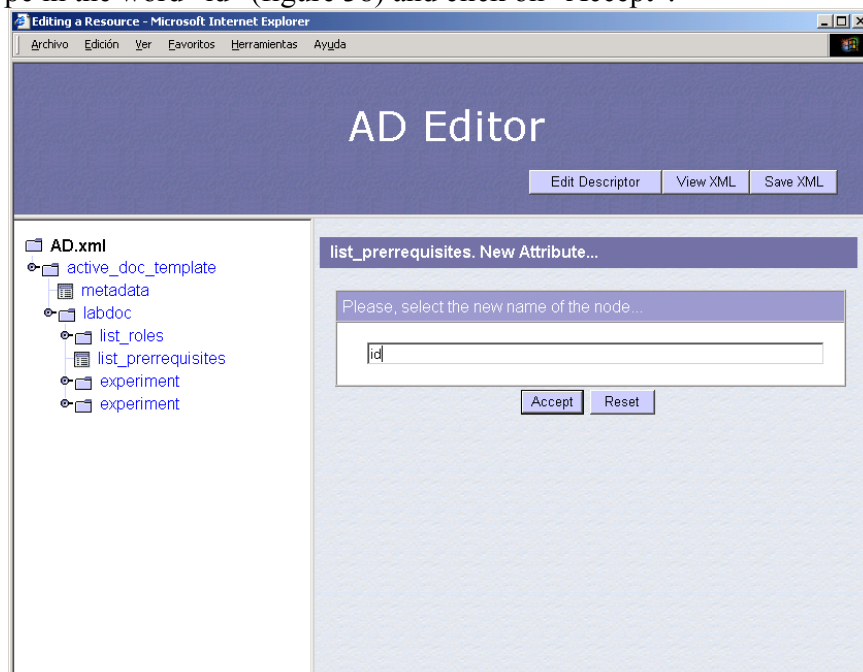


Figure 38. The *list_prerequisites* element needs an attribute *id* (identifier)

3. Go to the node again and select the attribute id in the table “Attributes” in the table on the right. Assign a value to the identifier (for example, “listPrerequisites1”) and click the button “Update” (figure 39).

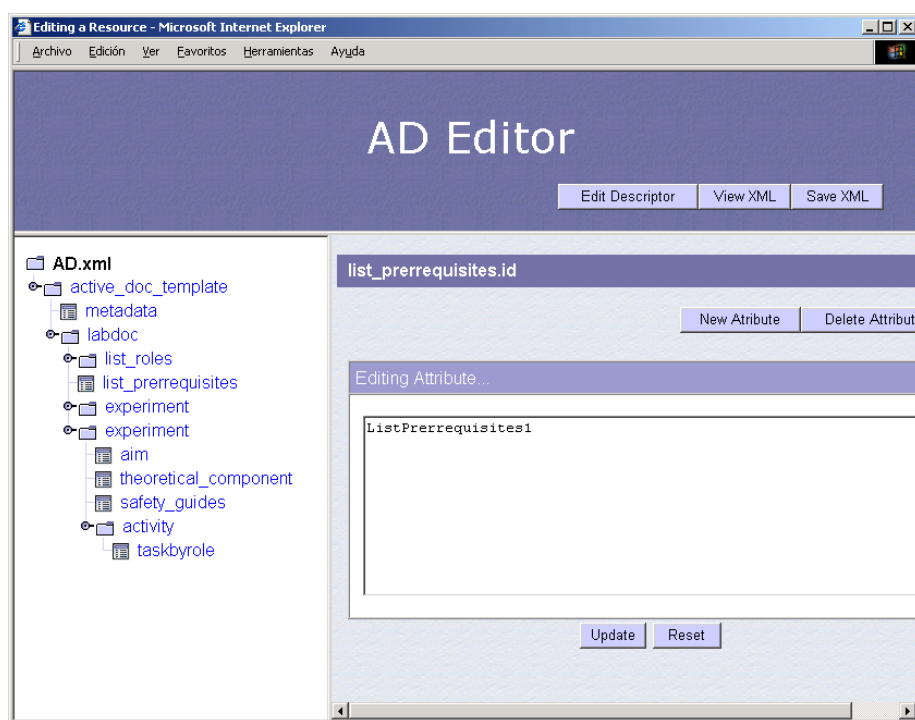


Figure 39. Filling in the identifier (*id* attribute) for the *list_prerequisites* element

4. Now we create our first prerequisite. We must go to the node *list_prerequisites* and click on the button “New Child”. On the text square, type “prerequisite” (figure 40) and click on “Accept”.

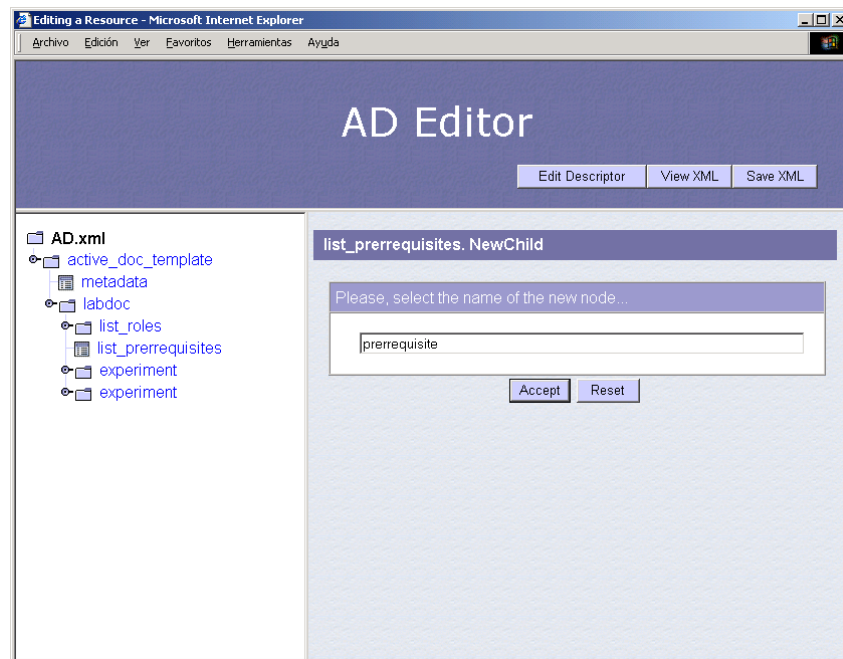


Figure 40. Adding a new prerequisite (*prerequisite* node)

5. Now three attributes must be created and given a value: a prerequisite identifier (*id*), a starting date (*date_in*) and an ending date (*date_out*). The process is similar to the one repeated in steps 2 through 3 so we will not repeat them again. As a result we should obtain the prerequisite just as it appears in figure 41.

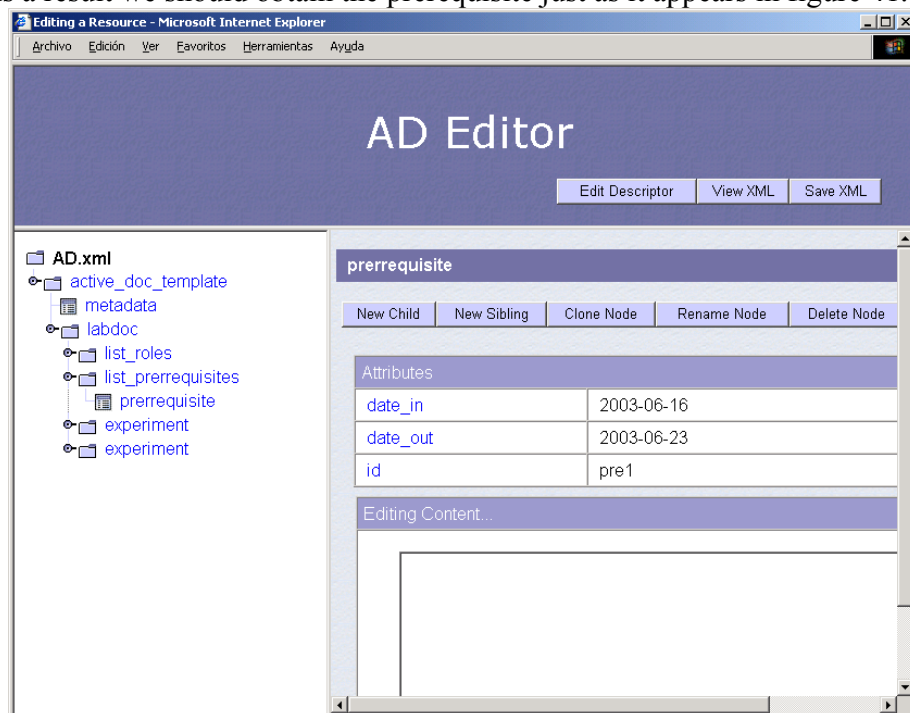


Figure 41. Attributes for defining a new prerequisite

6. Repeat the process for the second prerequisite (figure 42). This time the second prerequisite must have an identifier (id) as attributes, a reference to the experiment exp1 (ref with value exp_1) and an attribute type (type with value passed).

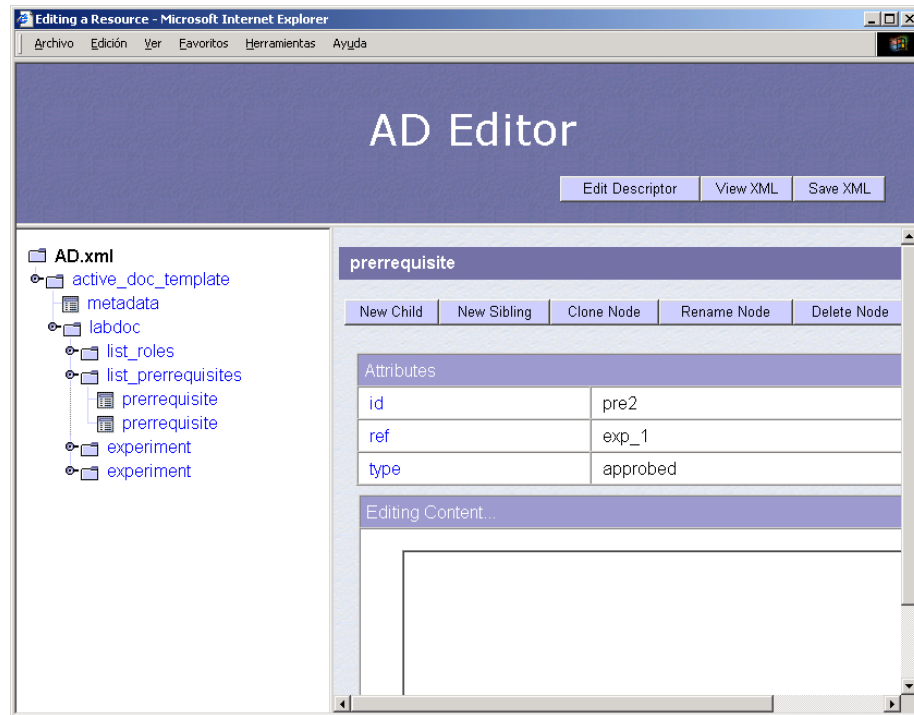


Figure 42. A new prerequisite and its attributes

7. Now we attach the prerequisite pre1 to the experiment exp_1 and the prerequisite pre2 to the experiment exp_2. To do so, we must add an attribute “prerequisite” (figure 43).

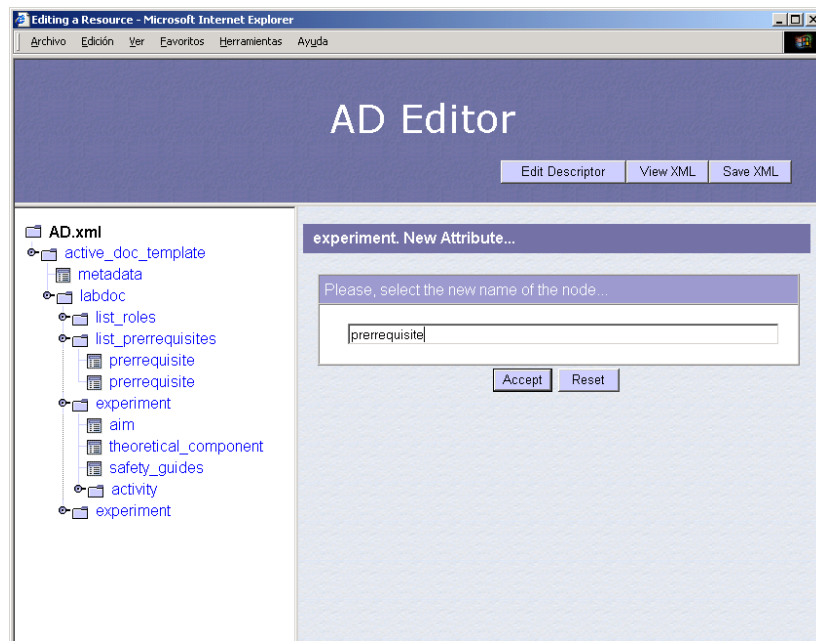


Figure 43. Adding a new prerequisite for the current experiment

8. And assign the value pre1 to the corresponding node to the experiment exp_1 (figure 44).

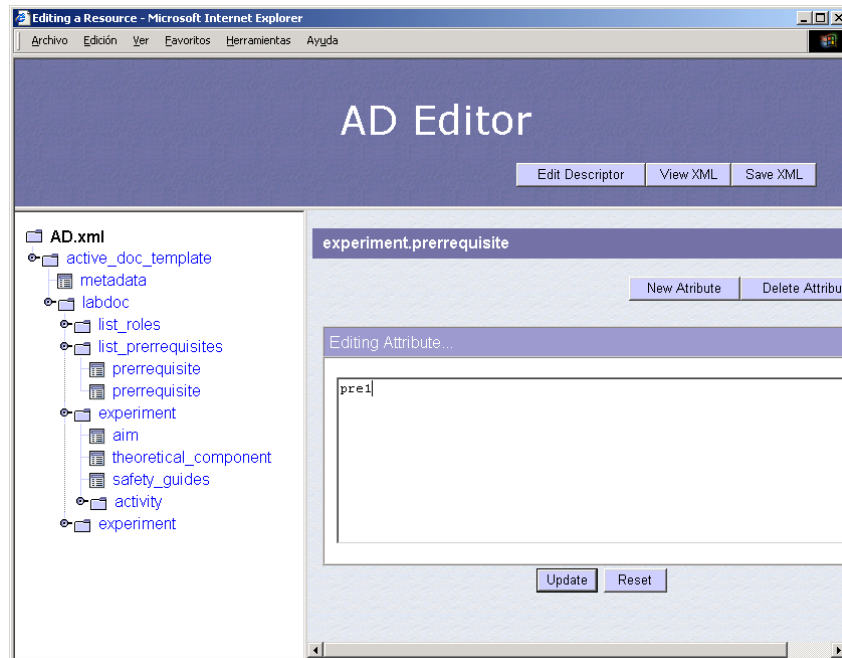


Figure 44. Identifying the experiment's prerequisite

9. This process must be repeated for the experiment exp_2 and the prerequisite pre2. This leaves the structure of the document just as it appears in figure 45.

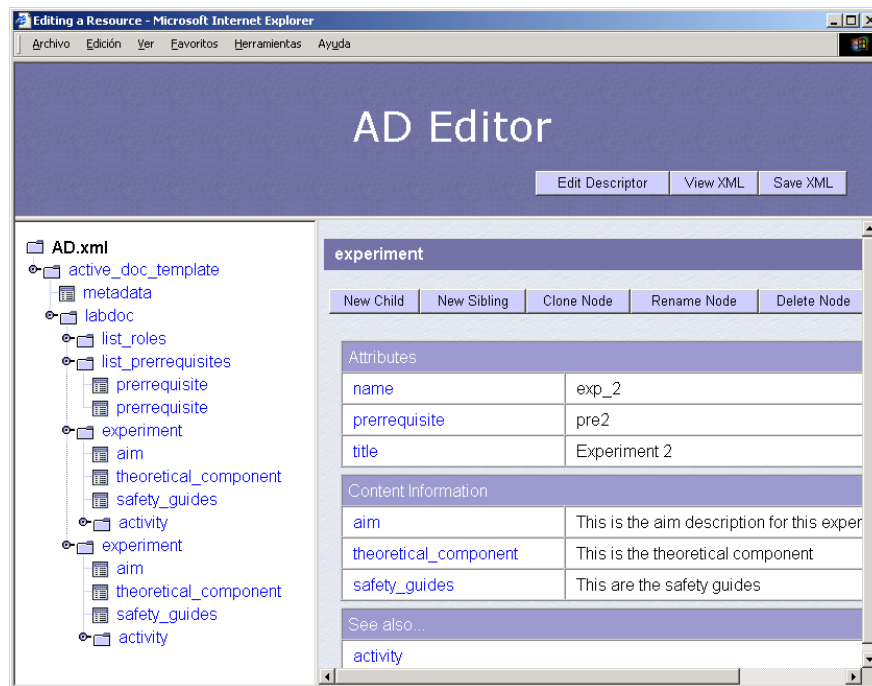


Figure 45. Attributes (including a *prerequisite*) and content for an *experiment*

4.1.2.4 Defining internal tools

The use of internal tools (ones which can not run without the AD System) in the environment can also require the appropriate configuration of the Active Document.

To use the following internal tools associated to a task we have to follow these steps:

➤ **editor:**

Make a reference to the resource whose attribute “id” within the resource file is “FormularioEditor”. So, we have to create an element of the type “resource_ref”, assign its attribute “id_ref” the value “FormularioEditor” and assign it the value we desire to the parameter “text” of the tool editor. This value will appear as header of the Editor.

The rest of the parameters that the label “resource_ref” has defined are optional. In this example we give them a value but in the case that we do not, the system will use the values it has defined by defect.

```
<resource_ref id="ref7" id_ref="FormularioEditor" display="insite" label="Text Editor">
  <parameter>
    <param name="text" value="Comments"/>
  </parameter>
</resource_ref>
```

➤ **tooltest:**

Make a reference to the resource whose attribute “id” within the resource file is “ToolTest”. For this we need to create an element of the type “resource_ref”, assign its attribute “id_ref” the value “ToolTest” and then assign the following parameters the value we would like:

- **QUESTIONS:** Will contain the question
- **ANSWERS:** Possible answers to the question must be specified in the following way:
<Id_de_respuesta>) Possible answer
Where <Id_de_respuesta> will be a letter from the alphabet assigned in ascending order depending on the possible answers that there are.
To separate the answers from each other we use the character “#”.
- **JUST:** Corrections that the professor wants to make must be specified separating each of the possible answers with the character “#”.
- **responses:** <Id_de_respuesta> the correct answers separated from each other with the character “#”.

The rest of the parameters that the label “resource_ref” identifies are optional. In this example we assign them a value but if no value is assigned, the system will use the values that are defined by defect.

```
<resource_ref id="ref6666" id_ref="ToolTest" display="insite" label="Select" height="300" width="400">
  <parameter>
    <param name="QUESTIONS" value="Question"/>
    <param name="ANSWERS" value="a) Answer 1.#b) Answer 2.#c) Answer 3.#d) Answer 4."/>
    <param name="JUST" value="False#Correct#False#Correct"/>
    <param name="responses" value="b#d"/>
  </parameter>
</resource_ref>
```

➤ **DrawTool:**

Make a reference whose attribute “id” within the file of resources is “Drawtool”. So, we must create a “resource_ref” element and assign its attribute “id_ref” the value “Drawtool”.

The rest of the parameters that the label “resource_ref” defines are optional. In this example we assign a value but if we didn’t, the system would use the values defined by defect.

```
<resource_ref id="EditorMAref1" id_ref="Drawtool" display="insite" label="Draw Editor" height="300" width="300"/>
```

Next we describe the process of tool reference that basically consists of associating the indicated task in a node “mediating_tools” where to refer the uses of each tool.
If we want to add a tool called “FormularioEditor” to the task “taskdef02” we must follow these steps:

1. Go to the node corresponding to the task that you want to vinculate the use of the tool (a node of the taskbyrole type). Click on the button “New Child” and type in the word “mediating_tools” in the text square that appears (figure 46). Then click on “Accept”.

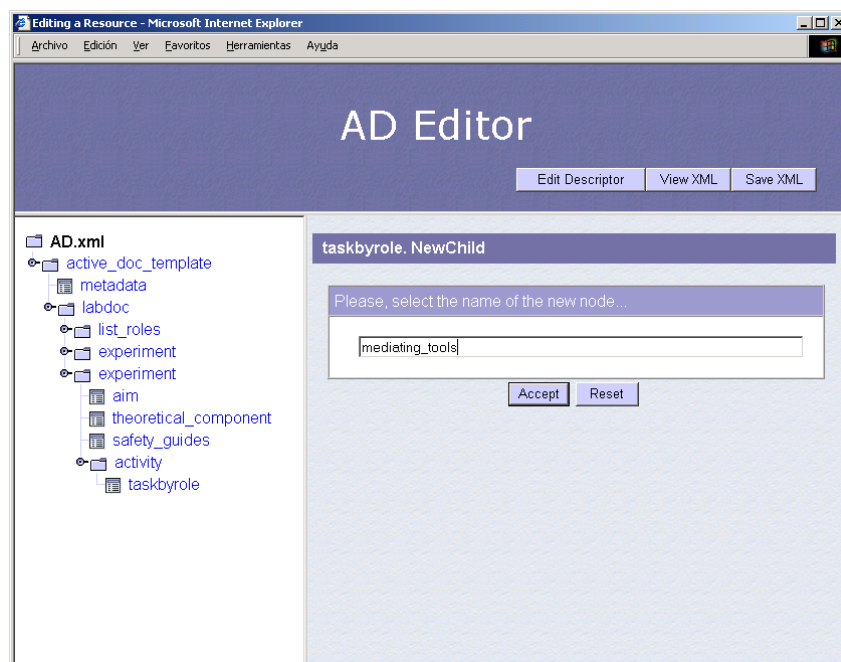


Figure 46. Adding *mediating_tools* to a *taskbyrole*

2. Go to the recently created node and repeat the previous process. In this case you must type the word “resource_ref” in the text square (figure 47).

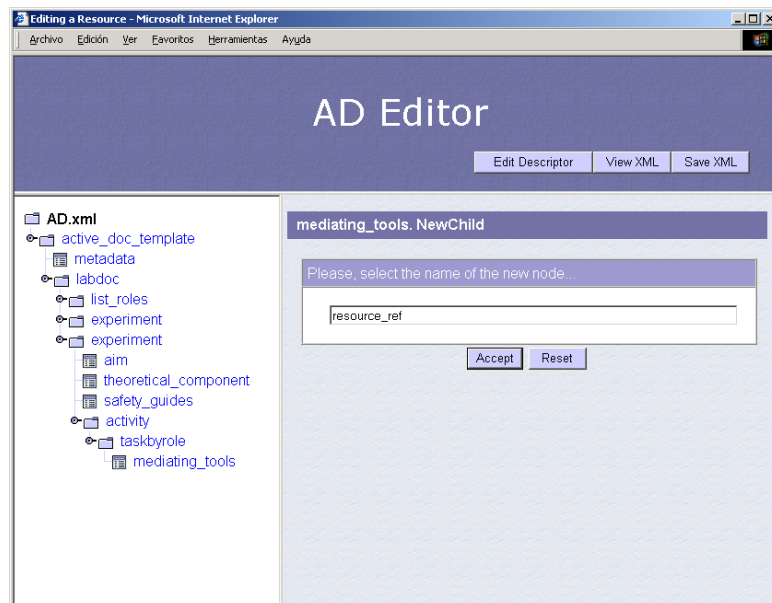


Figure 47. Adding a *resource_ref* to a *mediating_tools* node

3. Create a new name attribute id. Go to the recently created node and click on “New Attribute”. Type in the word “id” (figure 48) in the text square and click on “Accept”.

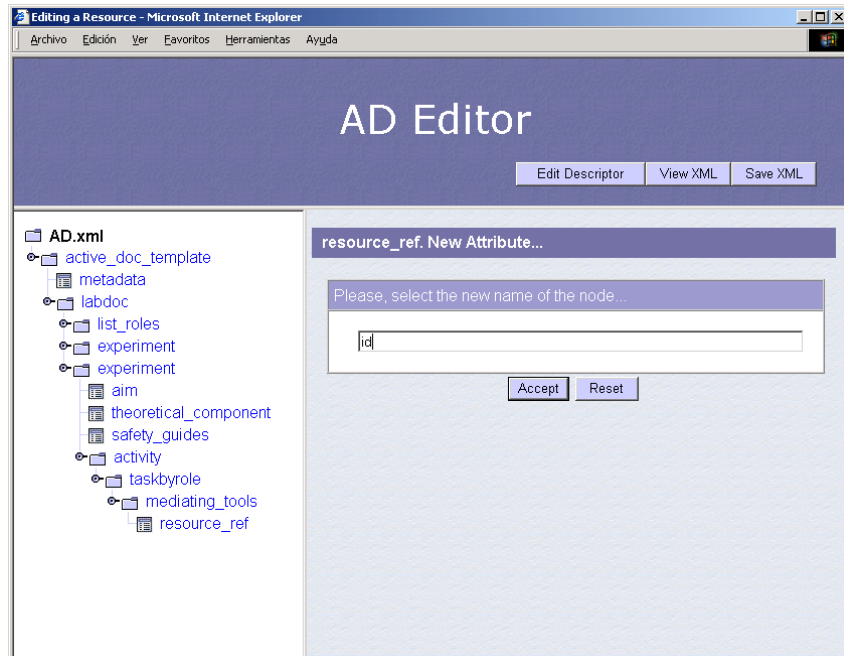


Figure 48. Adding an identifier (*id*) to a *resource_ref*

4. Assign a value to the attribute. To do so, follow the link id from the attributes table that appears in the right frame of the editor and assign it a value in the text area that appears (figure 49). Click on “Update”.

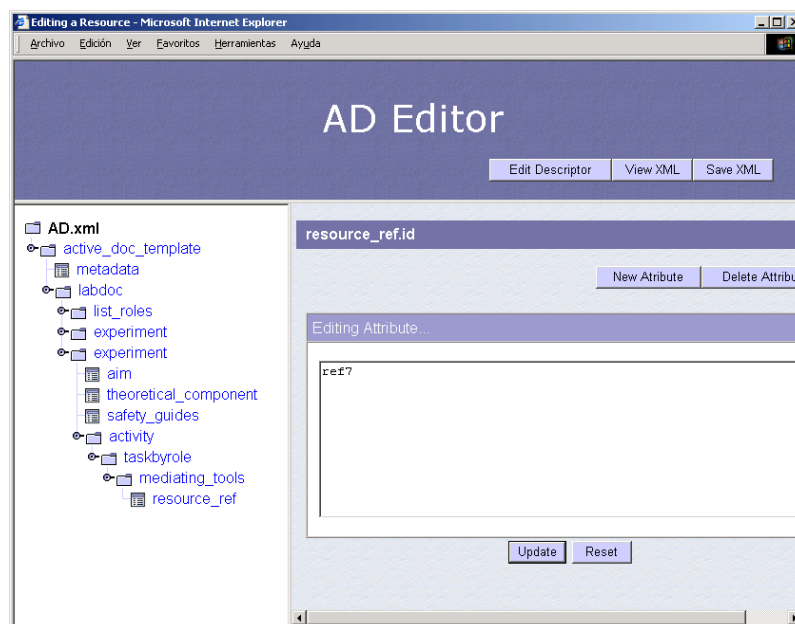


Figure 49. Filling in the *id*'s value for the *resource_ref*

5. Repeat steps 3 and 4 for the rest of the attributes: *id_ref*, *display*, *label*, *height* and *weight*, as shown in figure 50.

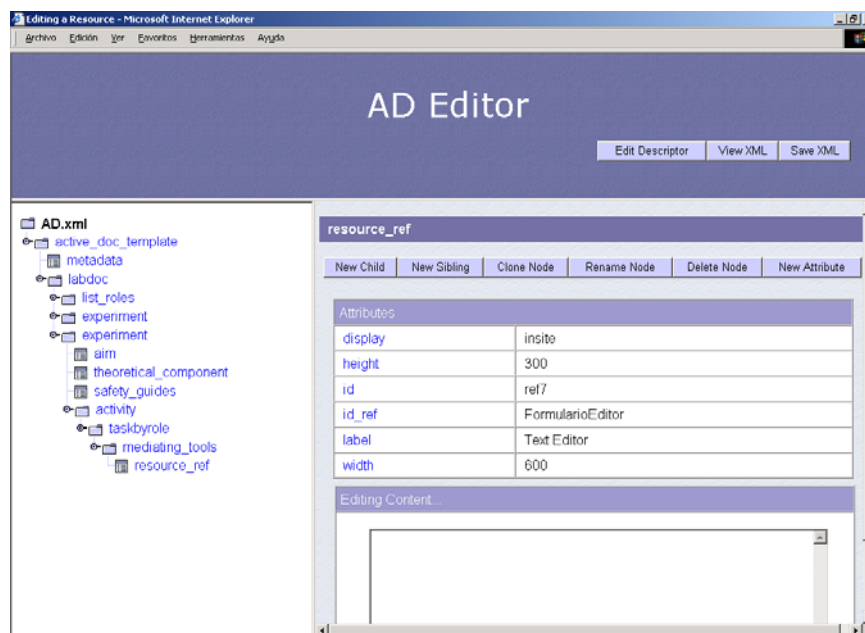


Figure 50. Attributes for a *resource_ref*

6. Now the tool parameters must be specified. To do so, go to the resource node `resource_ref` and create a new child with the name “parameter” by clicking on the button “New Child” (figure 51).

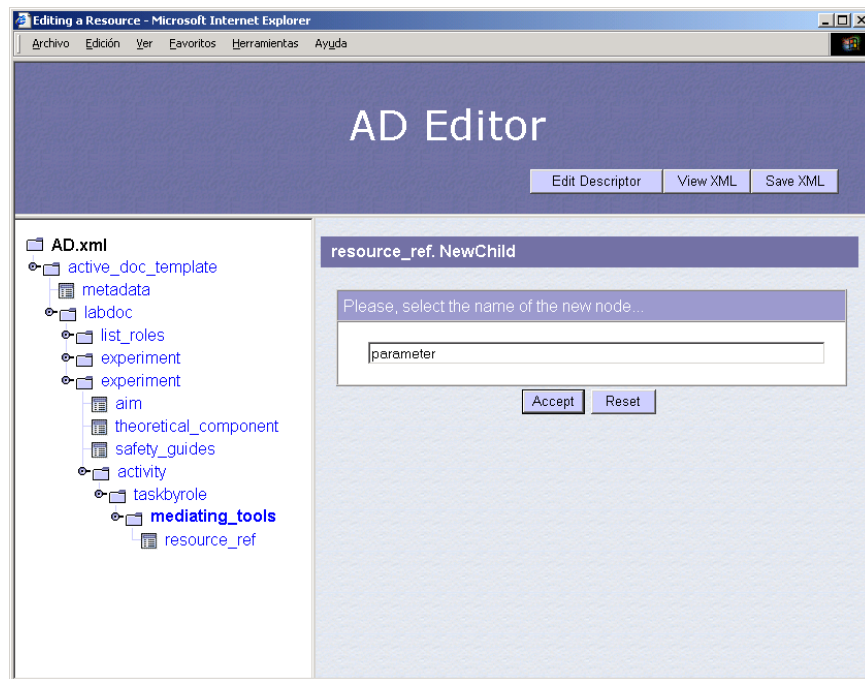


Figure 51. Adding a new *parameter* (a list of *param*) to a *resource_ref*

7. Go to this node and repeat the previous process creating this time a child node called “param” (figure 52).

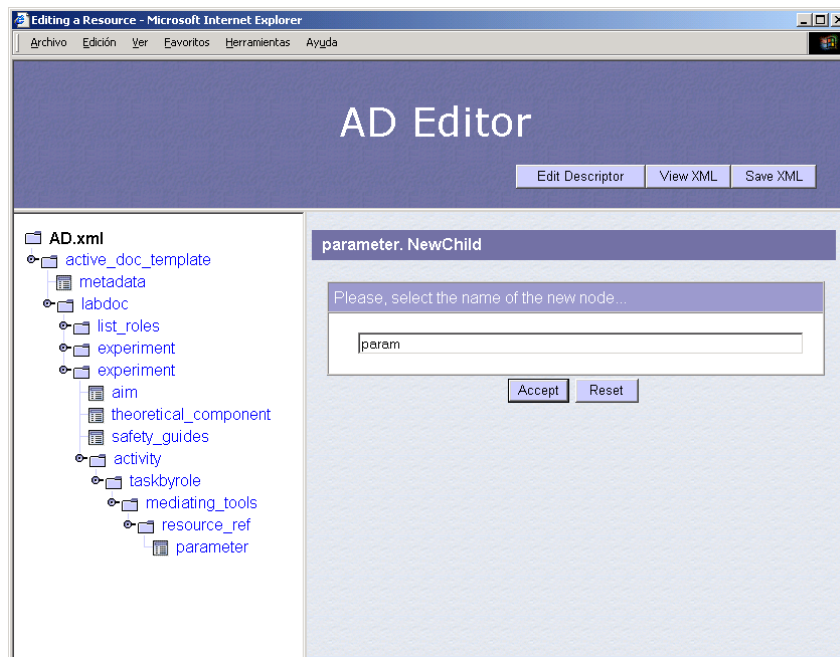


Figure 52. Adding a new *param* (an actual parameter) to a *parameter*(a *param* list)

8. Add an attribute called “name”. To do so, go to the node *param* and click on New Attribute. Then type in the word “name” in the text square (figure 53) and click Accept.

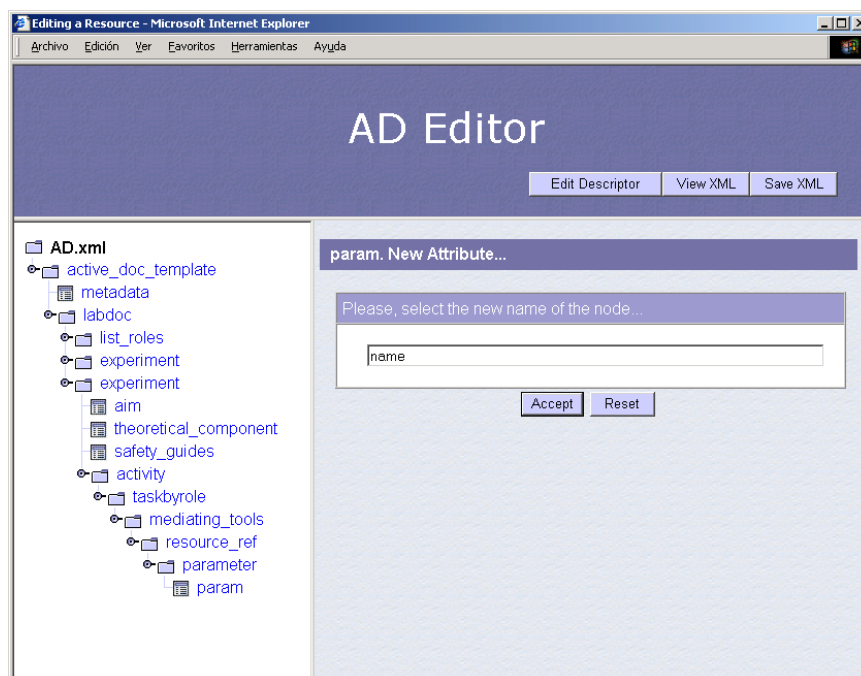


Figure 53. Adding the attribute *name* to a new *param*

9. Select the attribute in the Attributes table in said node and assign the name of the parameter (in this case it is “text”) as a value in the text area that appears (figure 54). Click the Update button.

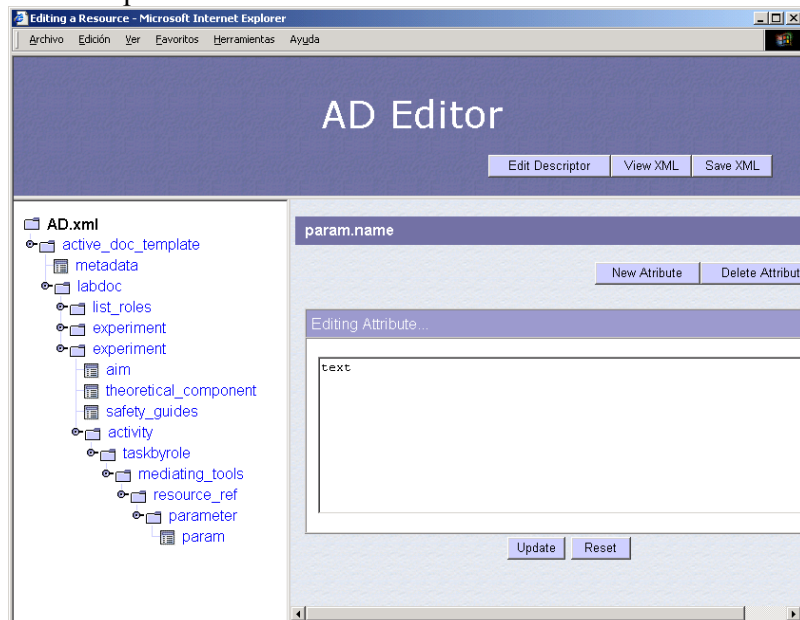


Figure 54. Naming the *param*: this parameter's name will be 'text'

4.1.2.5 Defining the correction tool

One special type of references of use that can be configured are those which refer to correction tools used by teachers (marker) to correct students. This type of reference requires a task with specific peculiarities about where to unfold. More specifically, it is necessary to create a structure of the task type (taskbyrole) whose attribute id is the same as the task it is correcting only the suffix “_m” is added. The only role admitted in this task is the role of the corrector (marker).

To associate the correction tool to a task, we must carry out the following steps:

- Define a new task in its own activity, whose id would be “<idTask>_m” that would contain the correction tool where <idTask> would be the id of the task we want to associate the correction tool to.
- Associate the correction task to the task that we want to correct through the attribute “markerTask”, to which we will assign the value the identifier of the task that contains the correction tool that will be “<idTask>_m”.

Example:

```
<taskbyrole id="taskdef01" name="Task 1" roles="student teacher" markerTask="taskdef01_m">
  Description of the Task 1
  <mediating_tools>
```

```

        <resource_ref id="EditorMAref1" id_ref="Drawtool" display="insite" label="Draw Editor"
height="300" width="300"/>
        <resource_ref id="ref7" id_ref="FormularioEditor" display="insite" label="Text Editor">
        <parameter>
        <param name="text" value="Comments"/>
        </parameter>
        </resource_ref>
        </mediating_tools>
    </taskbyrole>
    <taskbyrole id="taskdef01_m" name="correct" roles="marker">
    <mediating_tools>
        <resource_ref display="insite" height="500" id="ref666" id_ref="correction_tool" label=""
width="300">
        <parameter>
        <param name="just" value="Correct#Average#Wrong"/>
        <param name="values" value="10#5#0"/>
        </parameter>
        </resource_ref>
    </mediating_tools>
    </taskbyrole>

```

Changes in the community file:

- Modify the community file so that the task whose id is “<idTask>” is done by:
 - The students that want to carry out the task in the role of student
 - The teacher in the role of teacher

And the task that contains the correction tool is done by:

- The teacher in the role of teacher

Example:

```

<task_organisation task_name="taskdef01">
    <community_ref id="c01">
        <actor_ref id="test" role="student"/>
        <actor_ref id="marker" role="marker"/>
    </community_ref>
</task_organisation>
<task_organisation task_name="taskdef01_m">
    <community_ref id="c01">
        <actor_ref id="marker" role="marker"/>
    </community_ref>
</task_organisation>

```

Since the configuration is so similar to the previous case, we are not going to specify the configuration step by step. To see how tasks are configured and tools are referenced, the previous sections can be consulted. The final result should be like the one that appears in figure 55.

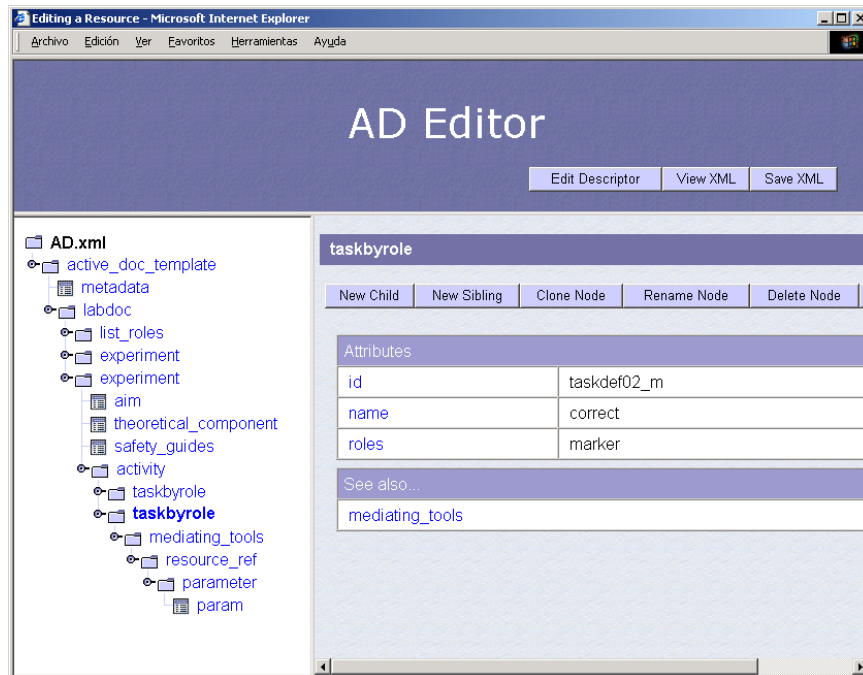


Figure 55. Attributes for a *taskbyrole*

It is also necessary to specify what task corrects the activity that we have just configured. In order to do that, we must specify an attribute “markerTask” in the task that is corrected for this task. The value of said attribute should correspond to the identifier of the correction task.

So, if we want our task to show correction task taskdef01_m we must correct the task taskdef01 and go to said node and add the attribute markerTask with value taskdef01_m. This is summarized in the following steps:

1. Go to the node taskbyrole whose identifier has the value taskdef01. Click on the button New Attribute. In the text square, write the word “markerTask” (figure 56) and click Accept.

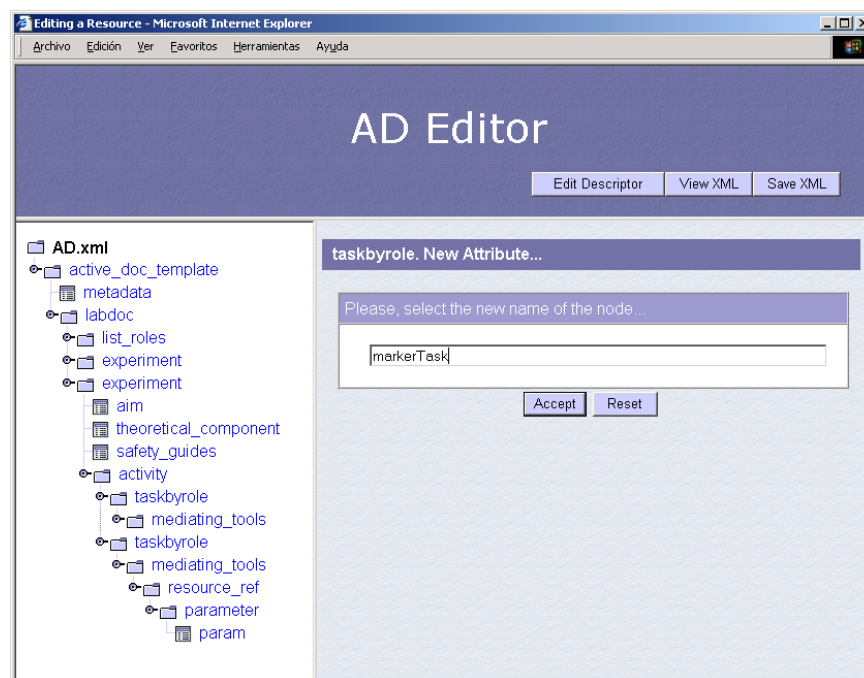


Figure 56. Adding a *markerTask* to a *taskbyrole*

2. Select the attribute from the table Attributes in the right side frame of the editor and type in the word `taskdef01_m` in the text area that appears (figure 57). Finally, click on Update.

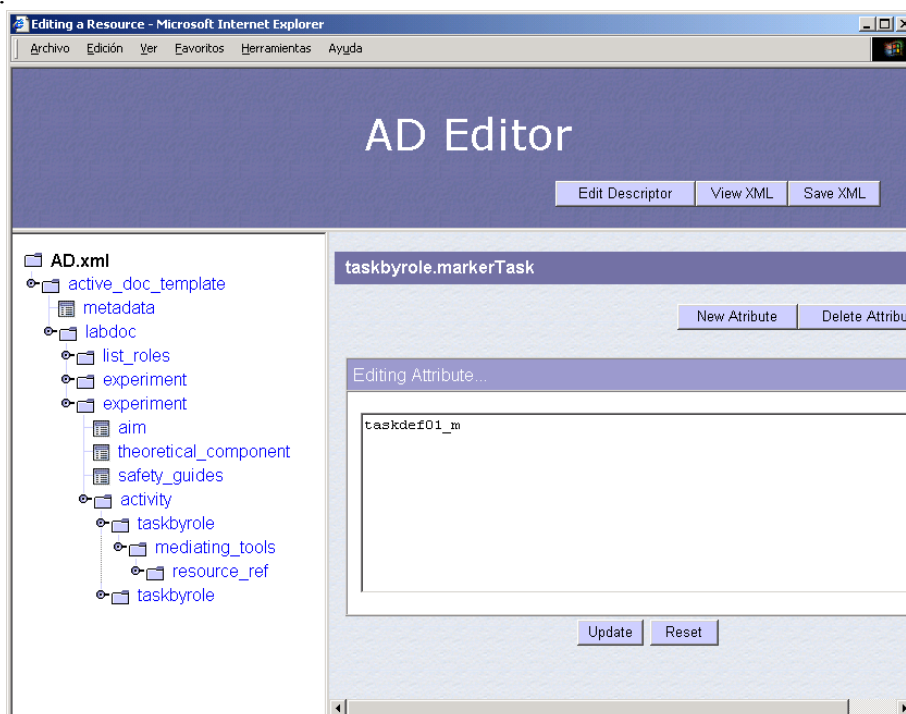


Figure 57. Filling in the *markerTask*'s name for the current *taskbyrole*

4.1.3 Definition of the resources

The Resources represent the support for a specific task to the members of a given community. For each task, a description of the available resources is given. This description is processed by the AD architecture in combination with the Description AD in order to relate the appropriate tasks to and tools. Performing the activities specified in an Active Document may either require or benefit the use of a number of resources, such as external document repositories or different types of tools.

4.1.3.1 Elements for defining a resource

Document	
	<document....>
Explanation	<p>Element: definition of document from IMS Content Packaging Specification v0.91</p> <p><u>Attributes:</u></p> <p>Id: defines a unique identifier used for identification and reference purposes, for example, when a resource is included in some experimental activity</p>
XML description	<pre><!ELEMENT document (file*)> <!ATTLIST document id ID #REQUIRED></pre>
Example	<pre><document id="d001"> <file uri="intro/introduction.html" format="HTML"/> <file uri="map.gif" format="image"/> <file uri="intro/safety.gif" format="image"/> <file uri="intro/description.html" format="HTML"/> </document></pre>
File	
	<file....>
Explanation	<p>Element: the actual file that holds the document</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> uri: location of the file that contains the actual fragment of the document format: limited list of accepted formats for the file
XML description	<pre><!ELEMENT file EMPTY> <!ATTLIST file uri CDATA #IMPLIED format (HTML PS PDF image) #REQUIRED</pre>
Example	<pre><file uri="map.gif" format="image"/></pre>
Operation	

	<operation....>
Explanation	<p>Element: operation configures permissions for the way in which a tool can be used</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ➤ Name: identification of the person in question ➤ Role: the category of the person which defines the way in which a tool can be used (for example, student, teacher, tutor)
XML description	<pre><!ELEMENT operation EMPTY> <!ATTLIST operation name NMTOKEN #REQUIRED role NMTOKEN #IMPLIED></pre>
Example	<operation name="monitor" role="tutor" />
Parameter	
	<parameter....>
Explanation	<p>Element: startup parameters required by a particular tool</p> <p><u>Attributes:</u> a set of name-value pairs for each parameter. Values are meant to be set at startup when required, so they are optional</p>
XML description	<pre><!ELEMENT parameter (param*)> <!ELEMENT param EMPTY> <!ATTLIST param name CDATA #REQUIRED value CDATA #IMPLIED></pre>
Example	<pre><parameter> <param name="portNo"/> <param name="bgimage"/> </parameter></pre>
Editor window	<i>See figure 69</i>
Resource	
	<resource....>
Explanation	<p>the basic resource definition which is either a document (text, image, etc) or some kind of tool</p> <p><u>Attributes²:</u></p> <ul style="list-style-type: none"> ○ id: identifier ○ use: In which context is the resource to be used ○ access_mode: how is the resource to be used
XML description	<pre><!ELEMENT resource (metadata?, (document tool))> <!ATTLIST resource id ID #REQUIRED use (general domain specific) #REQUIRED access_mode (single shareable) #IMPLIED></pre>

² Attributes typed in boldface are required for the element, as can be seen in their XML definition

Example	<pre> <resource id="document1" use="domain"> <metadata>In here we could put information which would help classify and facilitate search operations for this resource</metadata> <document id="d001"> <file uri="intro/introduction.html" format="HTML"/> <file uri="map.gif" format="image"/> <file uri="intro/safety.gif" format="image"/> <file uri="intro/description.html" format="HTML"/> </document> </resource> </pre>
Editor window	See figures 58 to 65
Tool	
	<tool....>
Explanation	<p>Element: the main tool definition</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ Id: defines a unique identifier used for identification and reference purposes, for example, when a resource is included in some experimental activity ○ Name: the title of the tool ○ Type: distinguishes between virtual and real tools ○ Uri: location of tool ○ Code: name of main class file in the case of Java programs ○ Mode: to be used either alone or in group ○ Format: what type of tool <ul style="list-style-type: none"> ▪ HTML: code intended to be pasted directly into the execution environment (in our case the ActiveDocument) ▪ Applet: a Java applet that can be run in the execution environment directly ▪ Application: an external application which would be run according to its definition and features ○ outcome_format - the type of result produced by the tool ○ outcome_uri - the location of the result in the case of external tools that do not actually directly return a result but require that a petition be made ○ icon - does the tool have an icon, and if so, its name
XML description	<pre> <!ELEMENT tool (workarea*, parameter?, operation*)> <!-- id ID #REQUIRED name CDATA #REQUIRED type (logical physical) #REQUIRED uri CDATA #IMPLIED code CDATA #IMPLIED </pre>

	mode (individual collaborative) #REQUIRED format (uri HTML applet application) #REQUIRED outcome_format (gif svg jpg plaintext word latex ps pdf uri) #IMPLIED outcome_uri CDATA #IMPLIED icon CDATA #IMPLIED
Example	<pre><tool id="DrawTool" name="Draw Editor" type="logical" uri="http://rigel.lsi.uned.es:4080/ad1_des/jsp/parser_jsp/Applets/Paint.jar" code="DrawTest.class" mode="individual" format="applet"> <workarea id="tool1_wa_01" outcome_type="UTF-8"/> </tool></pre>
Editon window	<i>See figures Figure 66 and 67</i>
Workarea	
	<workarea....>
Explanation	<p>Element: workarea defines the scope of application of a tool so that a particular tool can be used with different data</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> ○ Id: defines a unique identifier used for identification and reference purposes, for example, when a resource is included in some experimental activity ○ outcome_format: the type of result that is produced in the workarea
XML description	<pre><!ELEMENT workarea EMPTY> <!ATTLIST workarea id ID #REQUIRED outcome_type CDATA #REQUIRED</pre>
Example	<workarea id="tool1_wa_03" outcome_type="UTF-8"/>
Editor window	<i>See figure 68</i>

4.1.3.2 Editing a resource XML file

When editing the file RL.xml using the MetadataEditor, it is very important that it is done fulfilling the dtd called “AD_ResourceList_v2.dtd” because if it isn’t, the AD will not work.

The definition of the resources that will be used within the learning environment must be appropriately configurated editing the resource file. This file can be accessed by selecting the entrance RL.xml from the initial screen of the editor (figure 58).

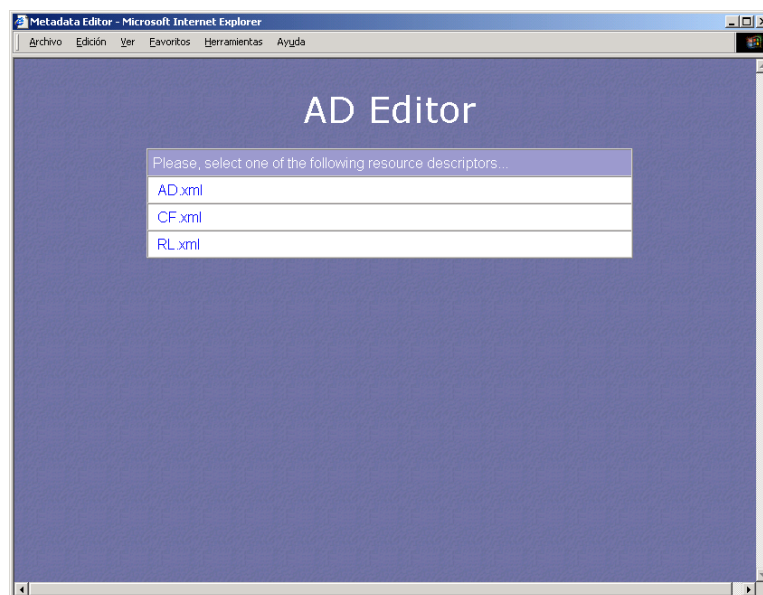


Figure 58. Selecting the resource file (RL.xml)

When editing this file we can see that its content mostly corresponds to the collection of resources defined to be used in this environment. By omission, the installation process creates a resource file that contains the most elemental work tools (figure 59).

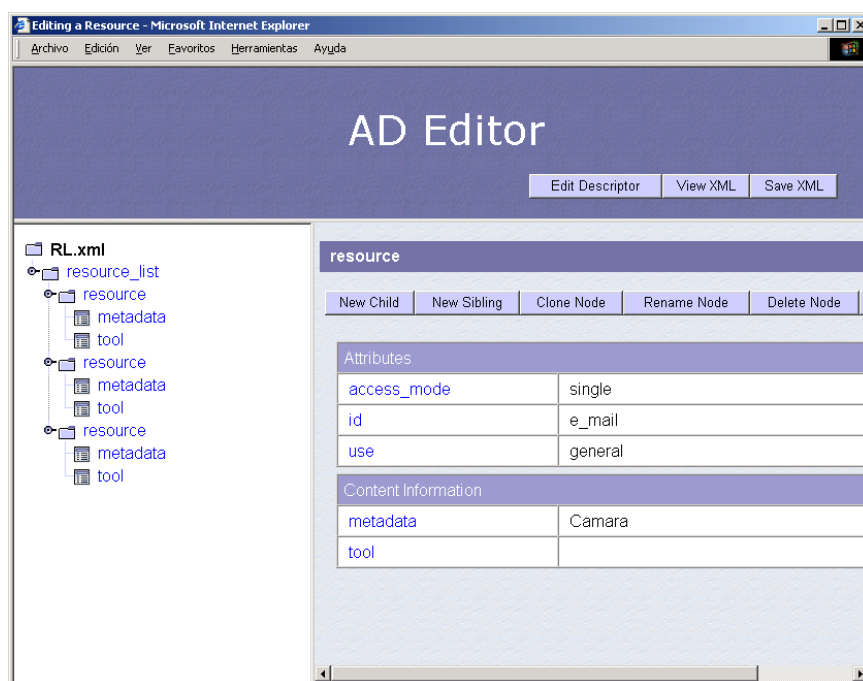


Figure 59. Default resource file with the basic tools included

Entering a new resource

Now we are going to explain the steps to follow to add a new resource to the list of those that already exist.

1. Go to the node `resource_list` and click on the button New Child. In the text area type in “resource” (figure 60) and click Accept.

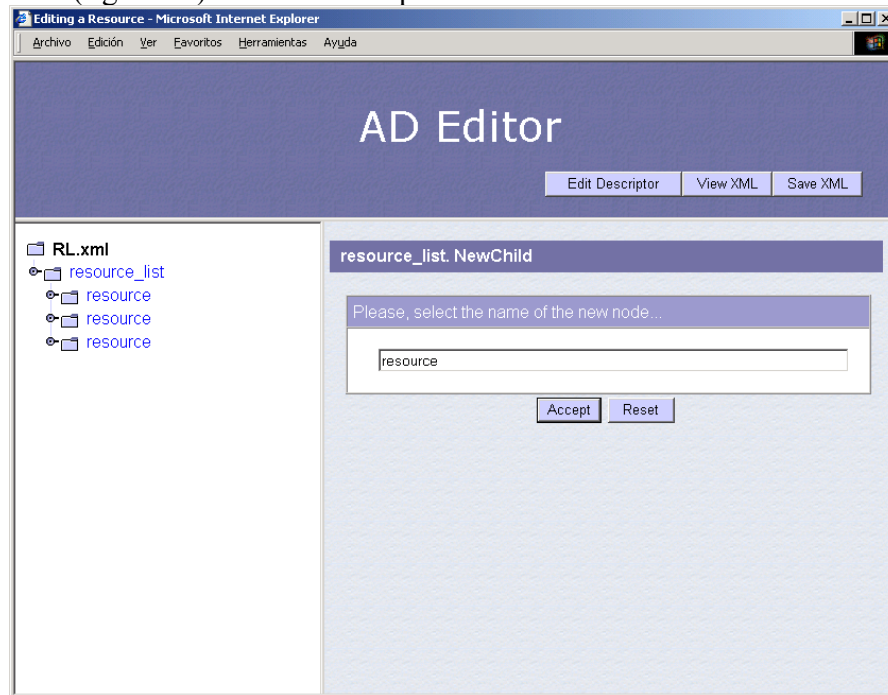


Figure 60. Adding a new *resource* to the *resource_list*

3. Go to the recently created node and repeat the previous process to create two child nodes with the names *metadata* and *tool* (figure 61).

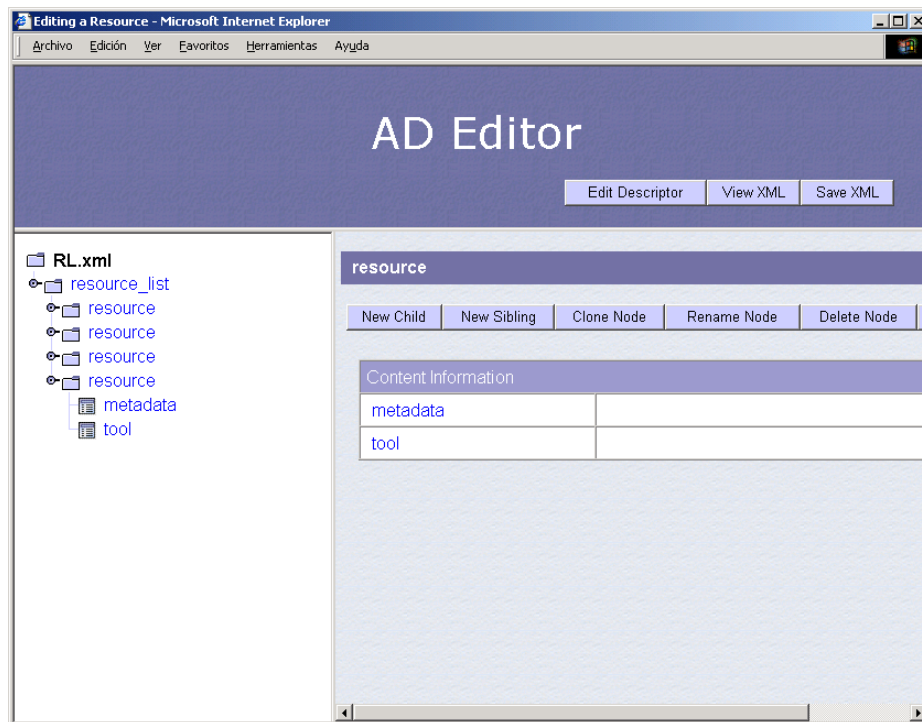


Figure 61. Adding *metadata* and a *tool* to a new *resource*

4. Create an attribute id. To do so, go to the node *resource* and select New Attribute. Write the word “id” in the text area (figure 62) and click on Accept. Repeat this process for the attributes “access_mode” and “use”.

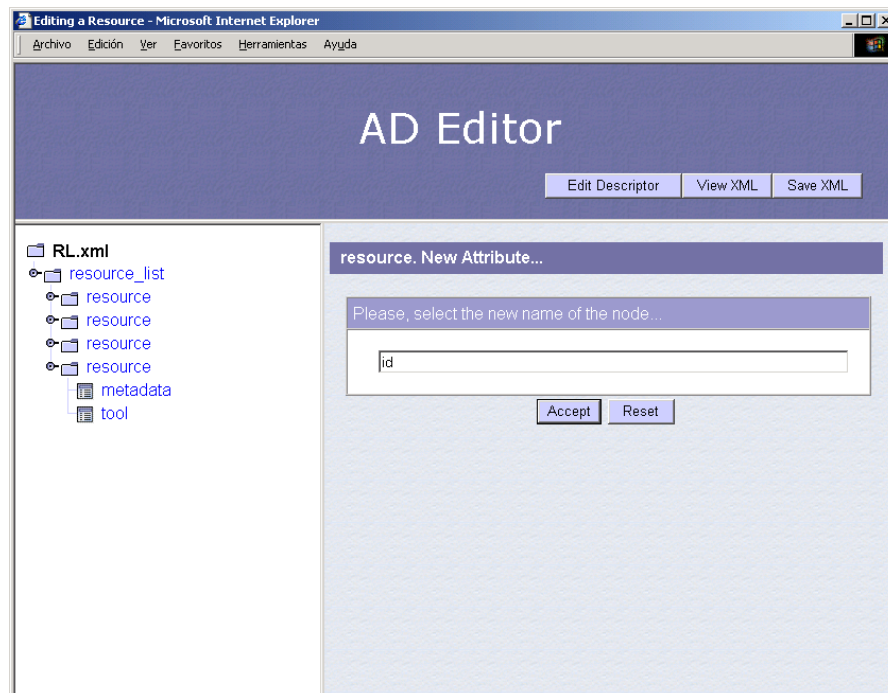


Figure 62. Adding the attribute *id*(identifier) to a new *resource*

5. Select an attribute from the table of attributes in the right side frame of the editor and assign it a value in the text area (figure 63). Then click on Update.

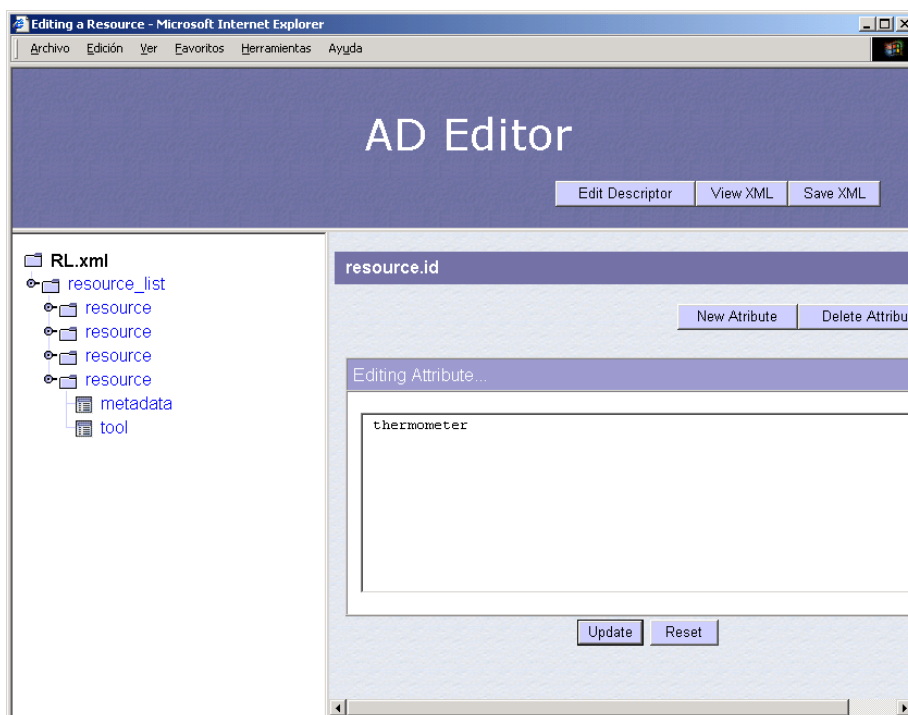


Figure 63. Filling in the name of the resource (*id*)

6. Repeat step 5 to give value to the other three attributes. As a result, you should obtain the resource configuration shown in figure 64.

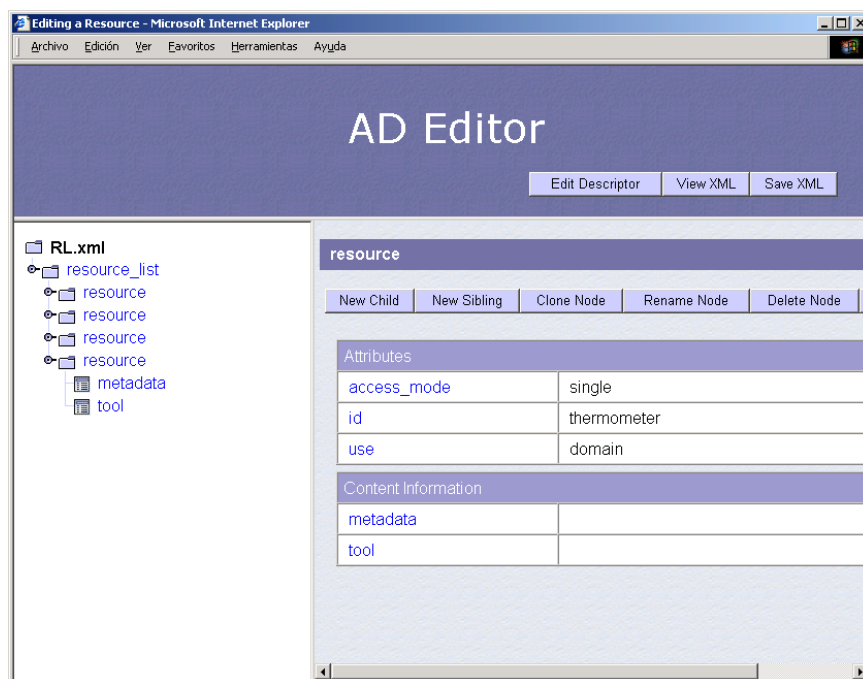


Figure 64. Attributes and content information for a new *resource*

7. Edit the field metadata. Go to the resource and follow the metadata link from the table Content Information located in the right hand frame. When the area of text shows up, add the metadata (figure 65) and click Update.

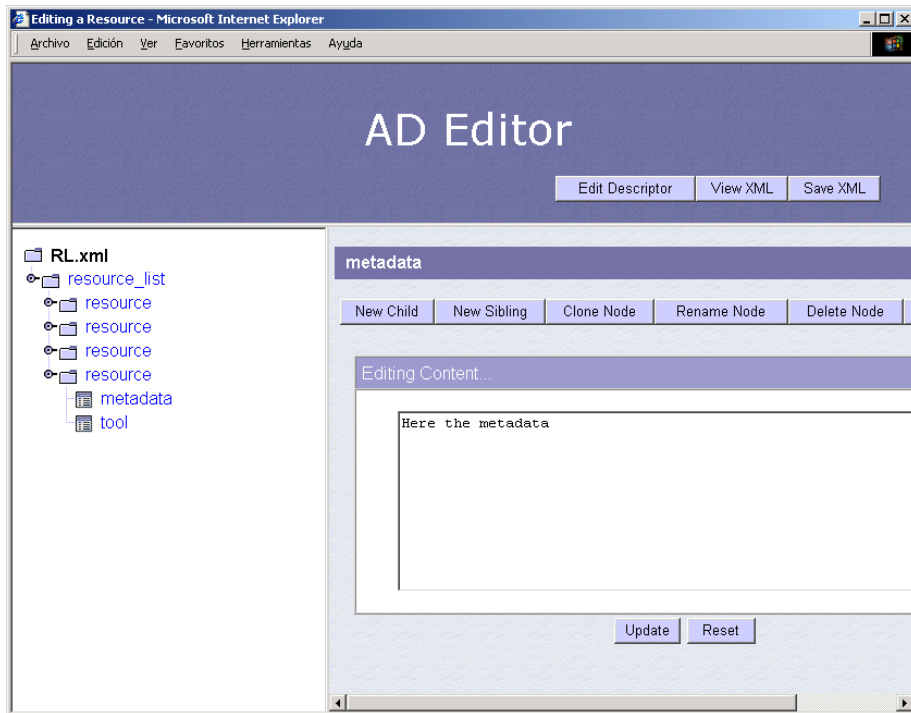


Figure 65. Adding the (standard) *metadata* for a new *resource*

8. Add the following attributes to the node *tool*: *id*, *name*, *type*, *uri*, *code*, *mode*, *format*. Go to the the node *tool* andn repeat steps 4 and 5. The result should be like the configuration of the resource that appears in Figure 66.

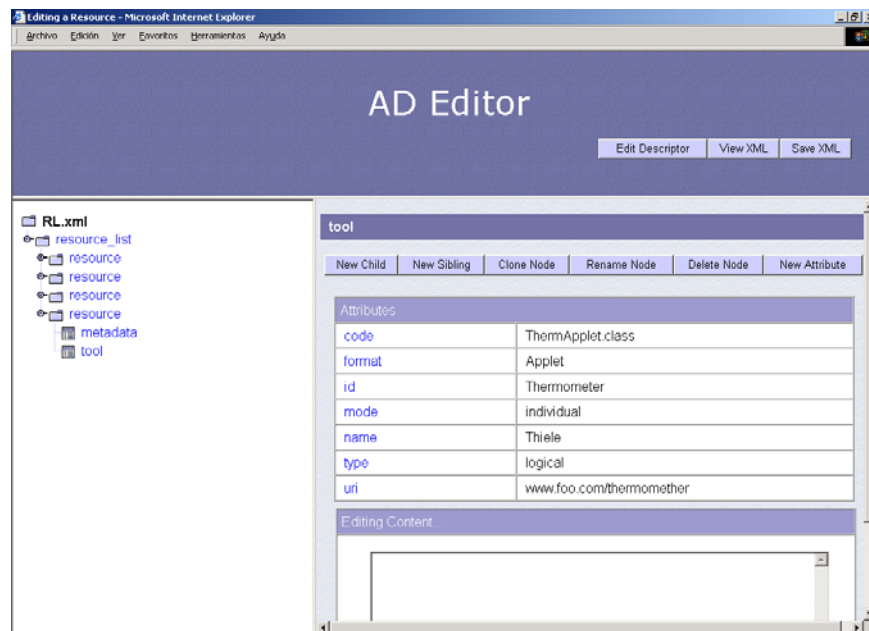


Figure 66. Inspecting the *tool*'s attributes

9. The two nodes which define the tool should be created. One that refers to the work area (workarea) and another that is relative to the parameters. The process is similar to the one in step 3. The results of the resource configuration appears in figure 67.

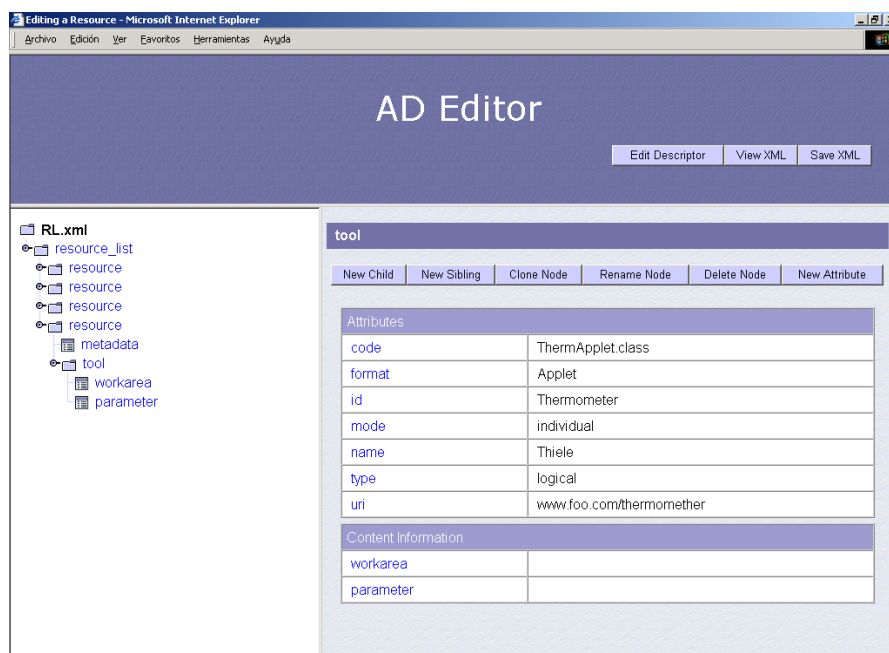


Figure 67. A *tool* requires a *workarea* and a *parameter* (a list of *params*)

10 . Go to workarea and add the attributes id and outcome_type. The process is similar to the one described in steps 4 and 5. The results appear in figure 68.

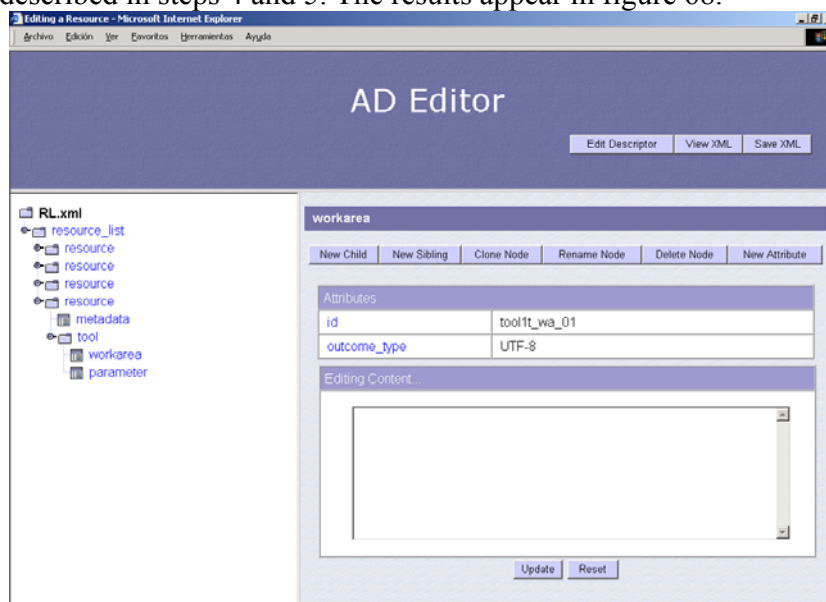


Figure 68. Attributes for defining a *workarea*

11. Go to parameter and create the nodes of the param type that constitute the tool parameters. The edition of one of them is shown in figure 69.

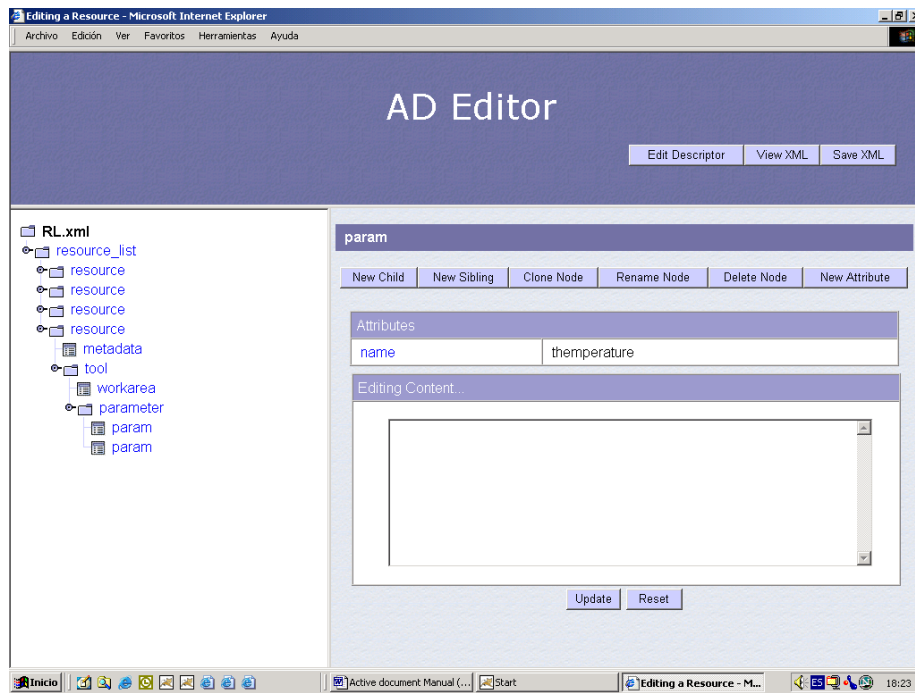


Figure 69. A *param* (an actual parameter named 'temperature')

4.1.4 Definition of the community

Within DiViLab, the community AD will be automatically generated by the Archimed LMS, as will be detailed in deliverable D7.7.

The Community AD represents the activity organization in order to describe the assignment of roles for a specific task to the members of a given community. For each activity, a description of the community involved is provided. This description is processed by the AD architecture in combination with the Description AD in order to relate the appropriate tasks and tools to the corresponding members of the community. The use of a separate XML structure for the community gives rise to two interesting mechanisms: firstly, communities can change during the development of the activity, thus allowing dynamic role assignments to be made (amongst other possibilities); and secondly, different Community AD can be combined with the same Description AD, providing a flexible mechanism for the re-use of the same division of labour description for a set of different working groups.

4.1.4.1 Elements for defining a community

Active_doc_template	
	<active_doc_template....>
Explanation	<p>Element: Active Document template for the course being defined</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o Name: the labdoc's name. It is not an identifier o Location: a literal describing the labdoc's location
XML description	<pre><!ELEMENT active_doc_template EMPTY> <!ATTLIST active_doc_template name NMTOKEN #IMPLIED location CDATA #IMPLIED></pre>
Example	<active_doc_template location="*****" name="ChemicalLab"/>
Activity_organisation	
	<activity_organisation....>
Explanation	<p>Element: activity_organisation, is the child unit to experiment_organisation within a community's division of labour. It is split into a non-null number of <i>task_organisation</i> elements</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o activity_name: the id and unique identifier of this activity
XML description	<pre><!ELEMENT activity_organisation (task_organisation+)> <!ATTLIST activity_organisation activity name NMTOKEN #REQUIRED></pre>
Example	<pre><activity_organisation activity_name="actividad_001"> <task_organisation task_name="taskdef01"> ... stuff cut for saving space see the following elements for details ... </task_organisation> <task_organisation task_name="taskdef01_m"> ... stuff cut for saving space see the following elements for details ... </task_organisation> <task_organisation task_name="taskdef02"> ... stuff cut for saving space see the following elements for details ... </task_organisation> </activity_organisation></pre>
Actor	
	<actor....>
Explanation	<p>Element: actor, each component of a community</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the actor's unique identifier
XML description	<pre><!ELEMENT actor EMPTY> <!ATTLIST actor id NMTOKEN #REQUIRED></pre>

Example	<code><actor id="usu1"/></code>
Editor window	<i>See figures 75 and 76</i>
Actor_ref	
	<code><actor_ref....></code>
Explanation	<p>Element: actor_ref, a reference to a community member indicating the actor's role in the task where this is involved</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the unique identifier of this actor o role: the one played by this actor for the current task
XML description	<pre> <!ELEMENT actor_ref EMPTY> <!--ATTLIST actor_ref id NMTOKEN #REQUIRED role NMTOKEN #IMPLIED--> </pre>
Example	<code><actor_ref id="test" role="student"/></code>
Editor window	<i>See figures 81 to 84</i>
Community	
	<code><community....></code>
Explanation	<p>Element: community, the list of the <i>actors</i> composing a community</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o id: the community identifier. o name: the community name
XML description	<pre> <!ELEMENT community (actor+)> <!--ATTLIST community id ID #REQUIRED name CDATA #IMPLIED--> </pre>
Example	<pre> <community id="c01" name="Group 1"> <actor id="test"/> <actor id="usu1"/> <actor id="usu2"/> </community> </pre>
Editor window	<i>See figures 72 to 74</i>
Community_ref	
	<code><community_ref....></code>
Explanation	<p>Element: a reference to an existent community</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o community_ref: the reference of the community
XML description	<pre> <!ELEMENT community_ref (actor_ref+)> <!--ATTLIST community_ref id IDREF #REQUIRED--> </pre>
Example	<pre> <community_ref id="c01"> <actor_ref id="test" role="student"/> <actor_ref id="usu1" role="student"/> <actor_ref id="usu2" role="student"/> <actor_ref id="marker" role="marker"/> </community_ref> </pre>
Editor window	<i>See figures 79 and 80</i>

Course	
	<course....>
Explanation	Element: course is the top element of the community configuration file and includes the Active Document template for the course being defined, a reference to an information model, the list of involved communities and a workplan
XML description	<!ELEMENT course (active_doc_template, information_model, list communities, workplan)>
Example	See the the following elements
Editor window	See figures 70 and 71
Experiment_organisation	
	<experiment_organisation....>
Explanation	Element: experiment_organisation, the topmost unit for a community's division of labour. It is split into a non-null number of <i>activity_organisation</i> <u>Attributes:</u> <ul style="list-style-type: none"> o experiment_name: the name of this experiment
XML description	<!ELEMENT experiment_organisation (activity_organisation+)> <!ATTLIST experiment_organisation experiment_name NMTOKEN #REQUIRED>
Example	<experiment_organisation experiment_name="exp_1"> <activity_organisation activity_name="actividad_001"> ... stuff cut for saving space see the following elements for details ... </activity_organisation> </experiment_organisation>
Information_model	
	<information_model....>
Explanation	Element: information model for the course being defined. It's used for documentation purposes <u>Attributes:</u> <ul style="list-style-type: none"> o BD_name: the information model's DataBase's name. o Location: the URI of the previous DB o Version: the DB's version
XML description	<!ELEMENT information_model EMPTY> <!ATTLIST information_model BD_name NMTOKEN #IMPLIED location CDATA #IMPLIED version CDATA #IMPLIED>
Example	<information_model BD_name="Chem01" location="http://sensei.lsi.uned.es/BD" version="mSQL"/>
List_communities	
	<list_communities....>
Explanation	Element: list_communities is a non empty set of community elements (<i>see</i>

	<i>community definition below)</i>
XML description	<!ELEMENT list_communities (community+)>
Example	<pre> <list_communities> <community id="c01" name="Group 1"> <actor id="test"/> <actor id="usu1"/> <actor id="usu2"/> </community> </list_communities> </pre>
Editor window	<i>See figure 71</i>
Task_organisation	
	<task_organisation...>
Explanation	<p>Element: task_organisation, is the child unit to activity_organisation within a community's division of labour (i.e., an activity_organisation element is composed of a number of task_organisation elements). It includes a non empty set of <i>community_ref</i> elements</p> <p><u>Attributes:</u></p> <ul style="list-style-type: none"> o task_name: the id and unique identifier of this task
XML description	<pre> <!ELEMENT task_organisation (community_ref+)> <!ATTLIST task_organisation task_name ID #REQUIRED> </pre>
Example	<pre> <task_organisation task_name="taskdef03"> <community_ref id="c01"> <actor_ref id="test" role="student"/> <actor_ref id="usu1" role="student"/> <actor_ref id="usu2" role="student"/> <actor_ref id="marker" role="marker"/> </community_ref> </task_organisation> </pre>
Editor window	<i>See figures 77 and 78</i>
Workplan	
	<workplan...>
Explanation	<p>Element: workplan; it reflects (see its subelements for details) the community's division of labour and consists of a non empty series of <i>experiment organisations</i></p>
XML description	<!ELEMENT workplan (experiment_organisation+)>
Example	<pre> <workplan> <experiment_organisation experiment_name="exp_1"> ... stuff cut for saving space see the following elements for details ... </experiment_organisation> </workplan> </pre>

4.1.4.2 Editing a community XML file

When editing the file CF.xml using the MetadataEditor it is very important to fulfil the dtd called “AD_Community_v23.dtd” because if it isn’t fulfilled, the AD will not work.

The last document that can be edited with the editor web is the communities document. This contains all the information having to do with user communities that have permission to use the learning environment. This file also indicates what tasks are done in what community. To edit this file we should select the entrance CF.xml from the initial menu of the editor (figure 70).

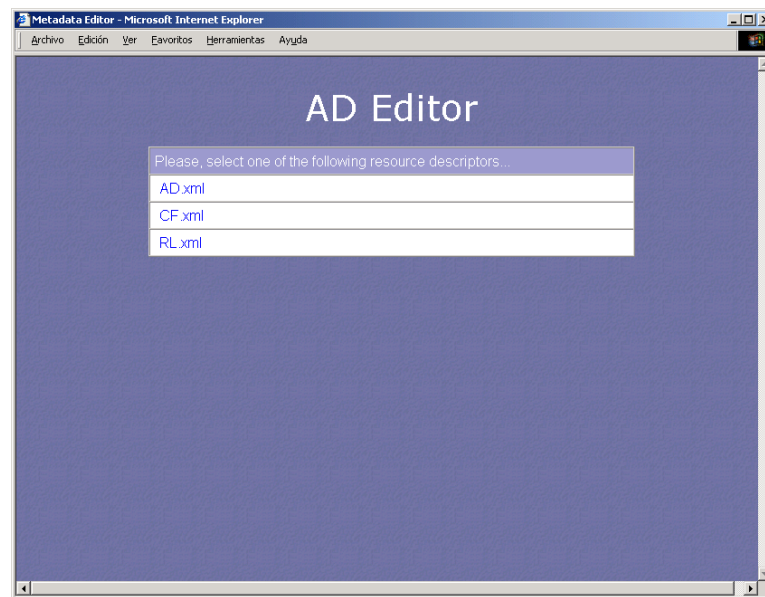


Figure 70. Choosing the community file (CF.xml)

As you can see, once we edit the communities file we can observe that there are two relevant nodes. One of them is list_communities that contains the collection of all of the communities. The other is workplan and it contains the organization of the experiments that allows them to be associated to communities (figure 71).

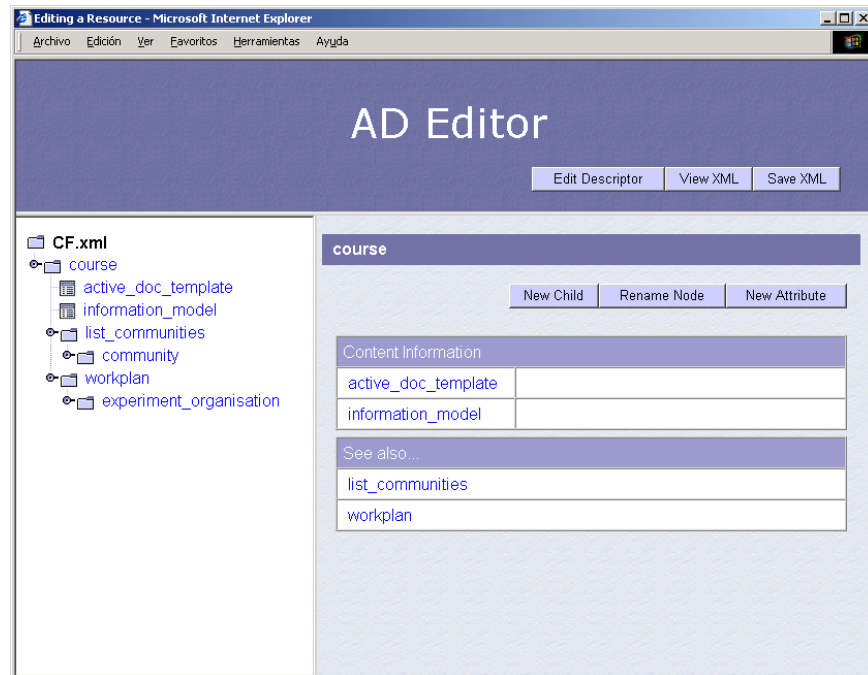


Figure 71. *course* structure showing the relevant nodes *list_communities* and *workplan*

So two configuration activities exist that deserve to be the cent of our attention. First of all, the addition of a new community to the learning atmosphere. Secondly, the assigning of an experiment for that community. In the next few lines we will describe these two situations in further detail.

Entering a new community

The entry of a new community is a process that requires the creation of the community and the addition of the actors that make it up in order to assign them their specific roles. Next we have the steps that must be followed:

1. Go to the node *list_communities* and click on the button *New Child*. Type the word “community” (figure 72) in the text square and click on *Accept*.

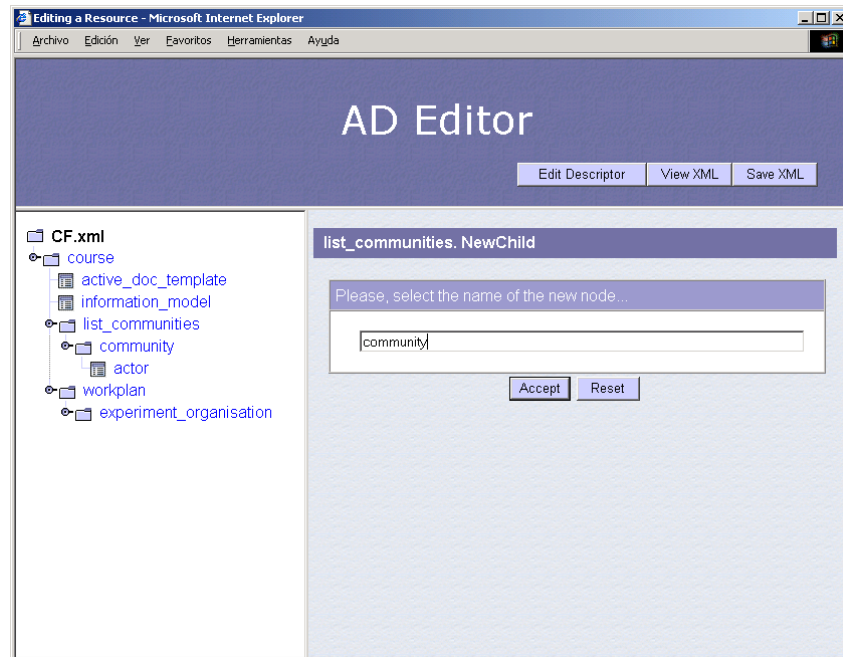


Figure 72. Adding a new *community* node to a *list_communities*

2. Go to the recently created node *community* and create two attributes: *id* and *name*. To do so, click on the button *New Attribute*, type the word "*id*" (figure 73) in the text square and click *Accept*. Repeat this process for the other attribute.

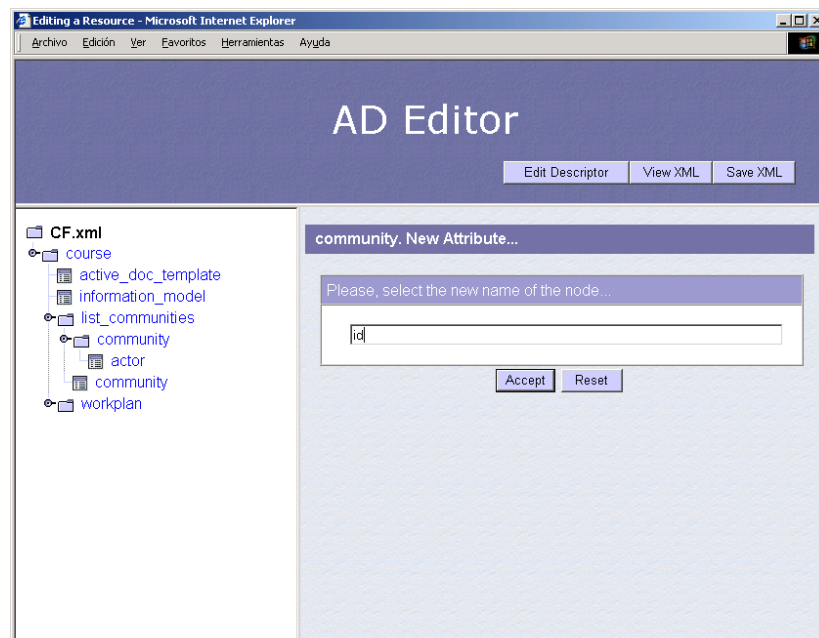


Figure 73. Adding the attribute *id* (identifier) to a *community*

3. Select an attribute from the table Attributes from the right side editor frame. In the text area, assign a value to the attribute (figure 74) and click on Update. Repeat this process with the other attribute.

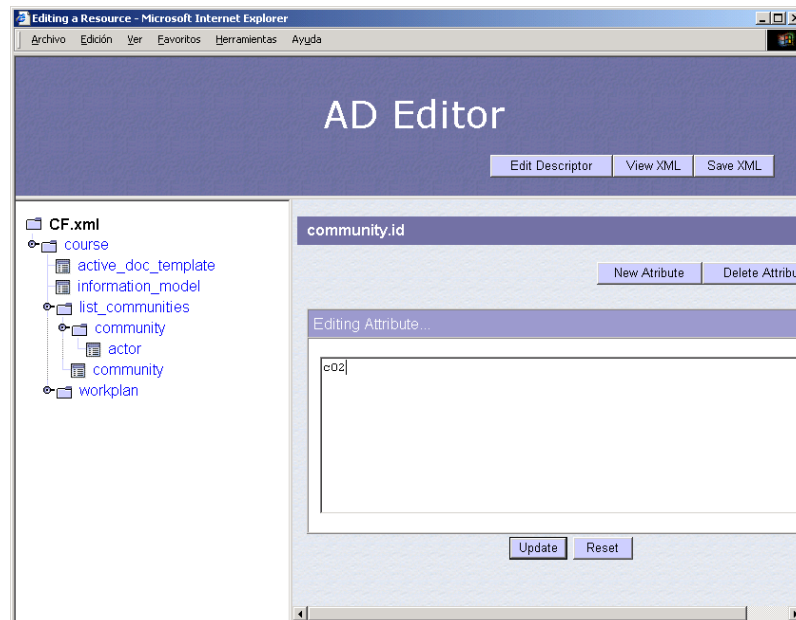


Figure 74. Filling in the value for the *community*'s *id*

4. Now create an actor associated with that community. Go to the node community that we are editing and click on the button New Child. In the text square, write the word actor (figure 75) and click on Accept.

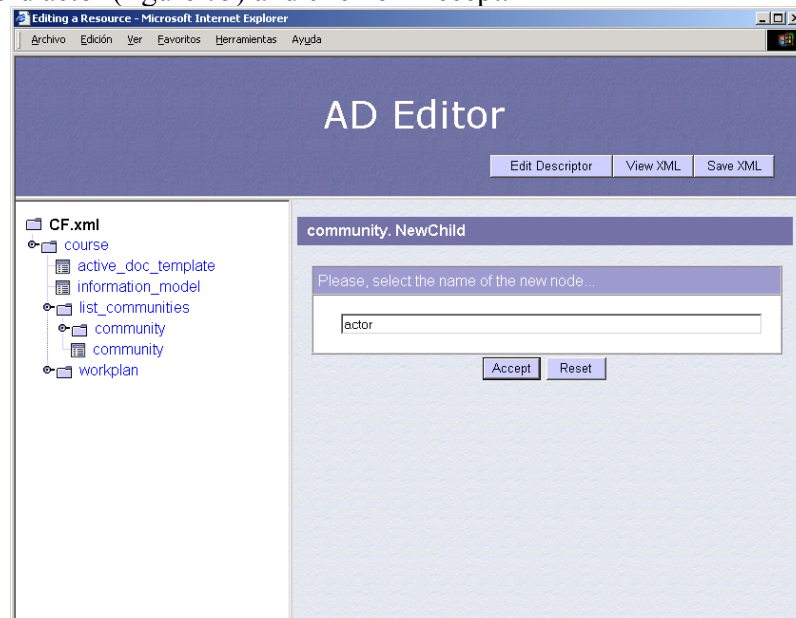


Figure 75. Adding a new *actor* to a *community*

5. Go to the node and add an attribute `id` that identifies the actor. Follow the steps described in points 2 and 3. Repeat the process described in steps 4 and 5 so that each actor has a community. That way we create a community of 3 users looking like figure 76.

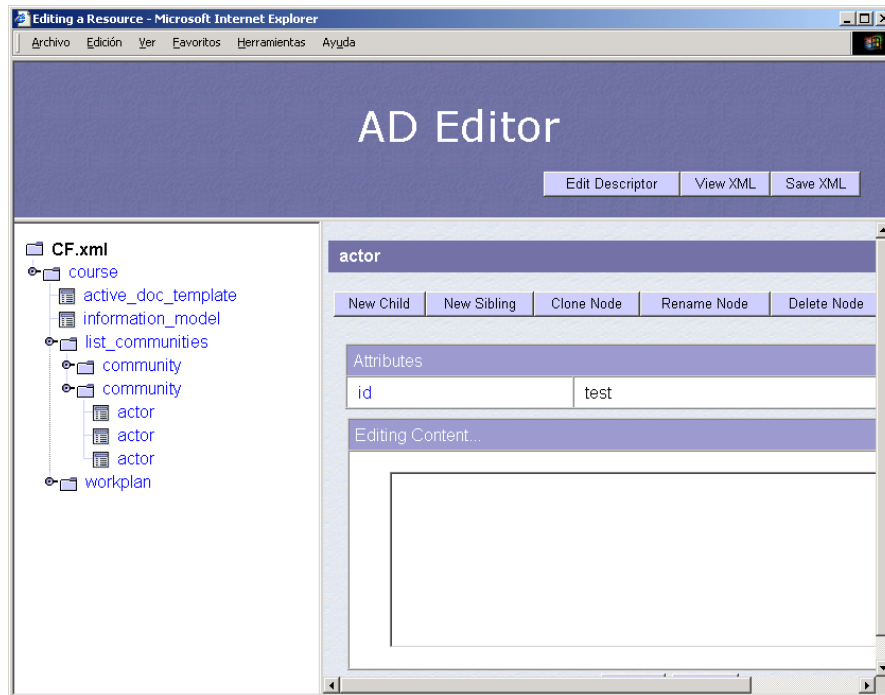


Figure 76. Adding the attribute `id` for the new *actor*

Assigning a task to a community

The other important configuration activity is assigning a task to a community. To show this aspect we will use the task that is defined by omission in the community document and we will add the one we created in the previous step. As shown in figure 77, the community `c01` has the task being edited assigned since a reference exists to that community as a child node.

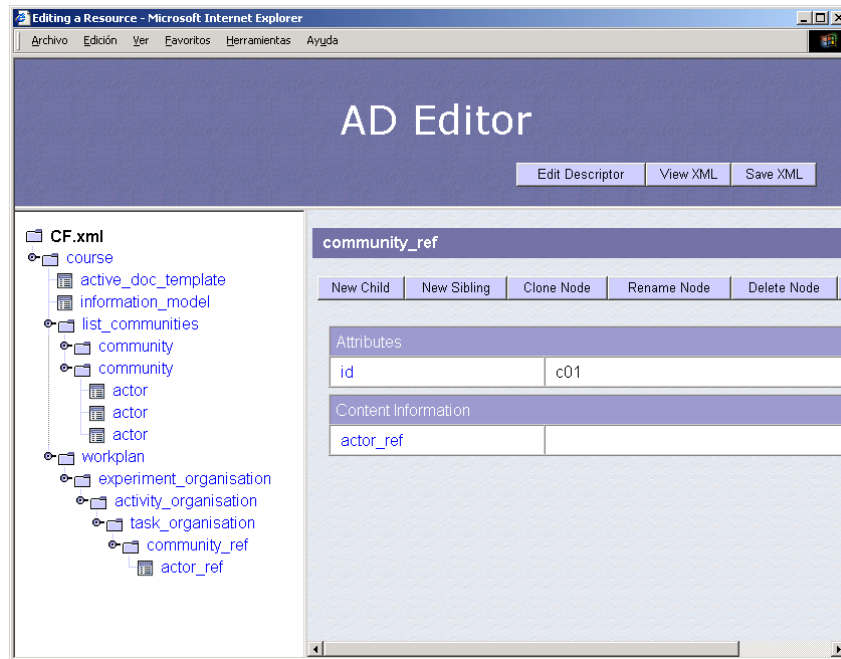


Figure 77. A *community_ref* including an *actor_ref*

We will be discovering the process to work with the community c02. So, these are the steps to be followed:

1. Go to the node *task_organization* that identifies the task that you want to be associated with the community. Click on the button *New Child* and write the word “community_ref” in the text square that appears (figure 78). Then click the button *Accept*.

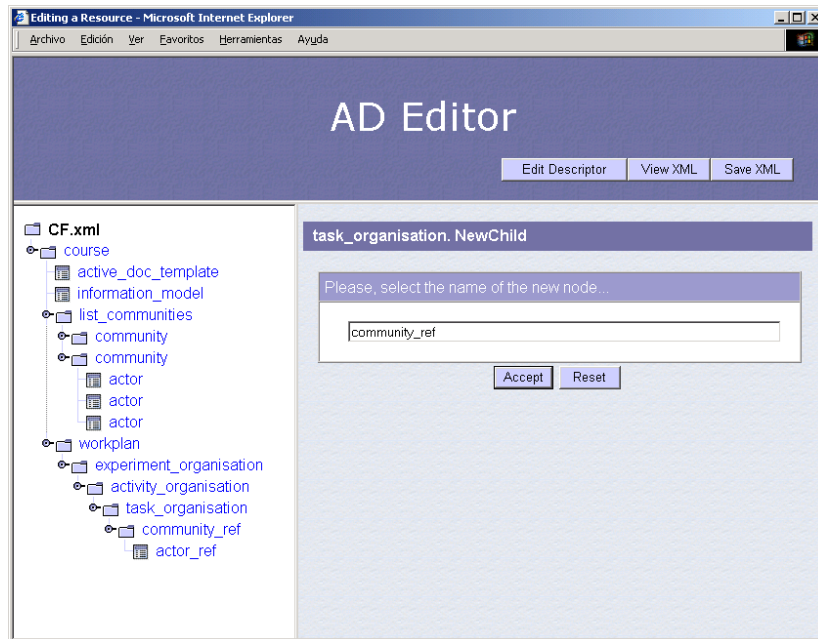


Figure 78. Assigning a community (by means of *community_ref*) to a *task*

2. Go to the recently created node *community_ref* . Add an attribute *id*. Click on New Attribute and type in “id” in the text square that appears (figure 79). Then click on Accept.

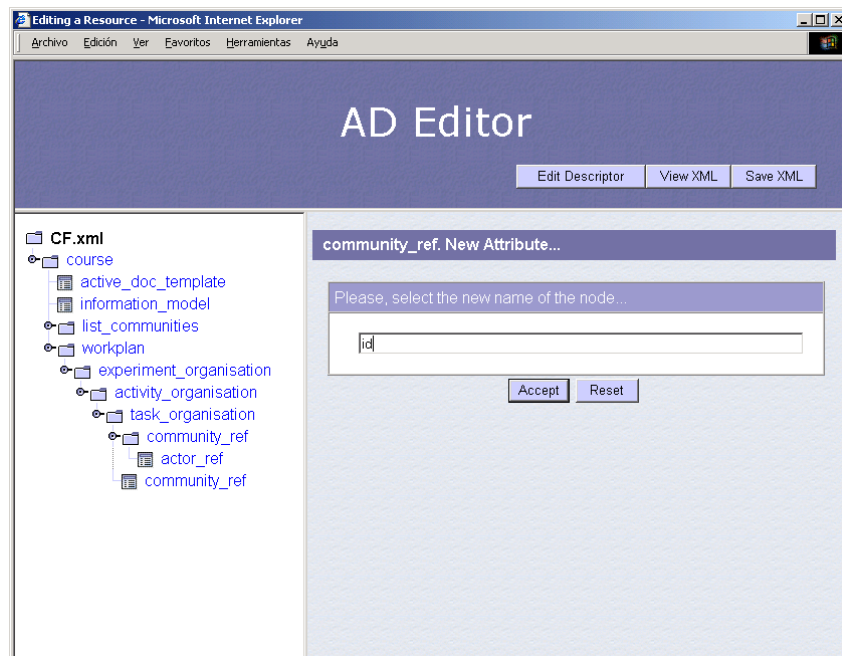


Figure 79. Adding the attribute *id* to a *community_ref*

3. Select said attribute in the table of attributes and assign it the value of the community you want that task to be associated with. In our example it is community c02 (figure 80). Then click on Update.

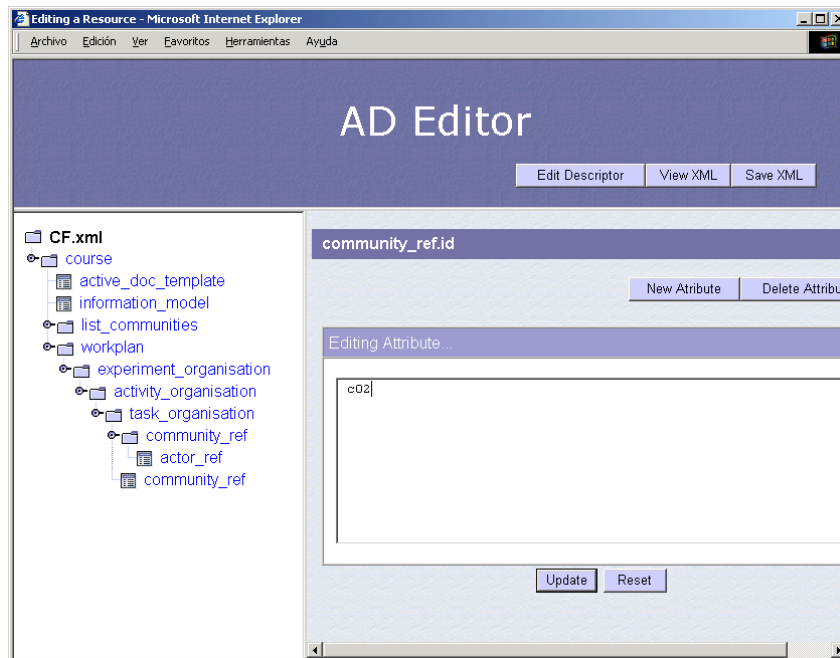


Figure 80. Filling in the community's *id* value for the *community_ref*

4. Now we must specify the member of the community that is in charge of developing the activity. We must create a reference to an actor from that same community. Go to the node *community_ref* and click on the button New Child. Type "actor_ref" in the text square that appears (figure 81) and click on the button Accept.

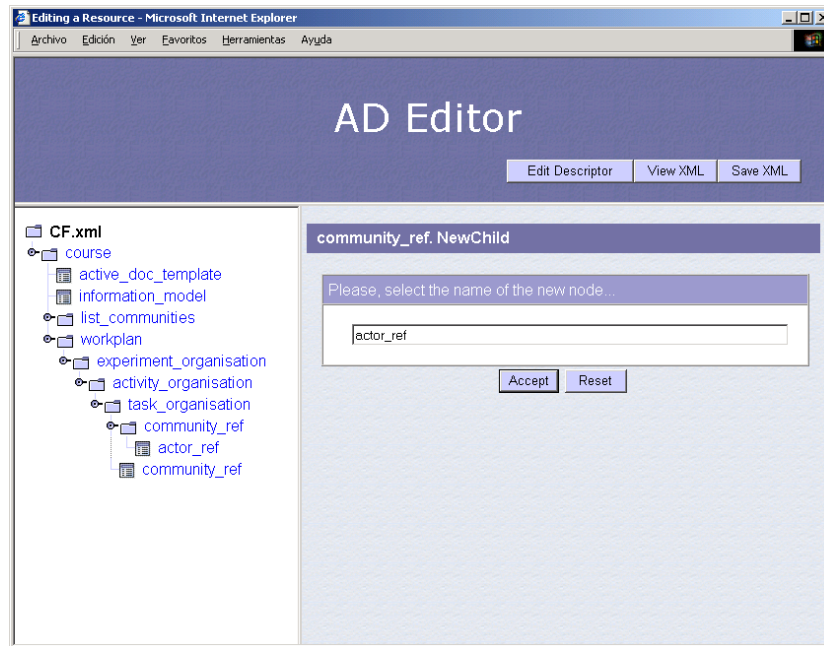


Figure 81. Adding the actor who will be responsible for this community.

5. Select the recently created node `actor_ref` and create two attributes: `id` and `role`. Click on the button `New Attribute` and write the word “`id`” (figure 82) in the text square. Repeat the process with the other attribute.

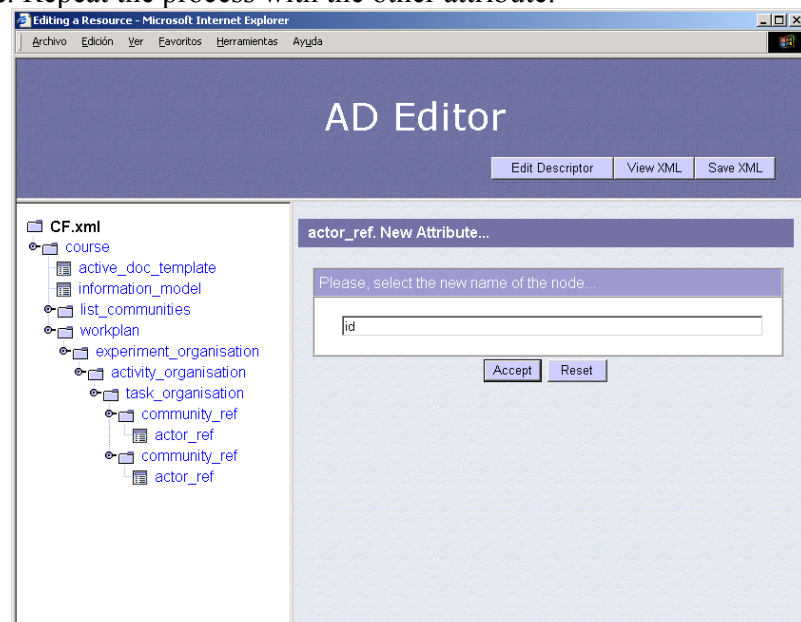


Figure 82. Creating the attribute `id` for the new *actor*

6. Assign value to these attributes. In our case `id` must contain the value `test` and `role` the value `student`. Go to the node `actor_ref` and follow the link of an

attribute. Write the value of the attribute in the text area (figure 83) and click on Update.

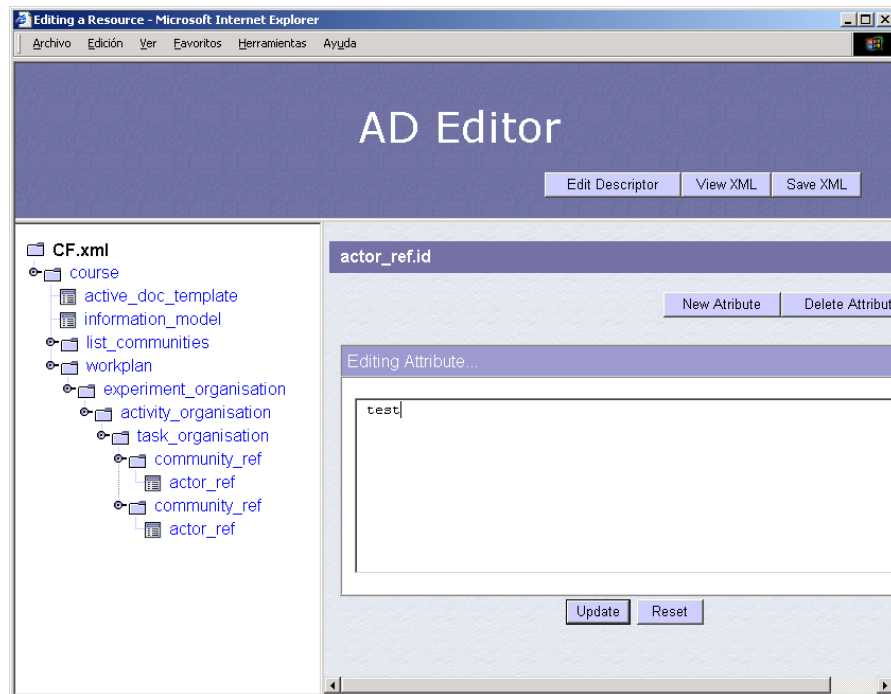


Figure 83. Filling in the value of the *actor_ref*'s *id* attribute

7. We should get an actor node like the one in figure 84 as a result.

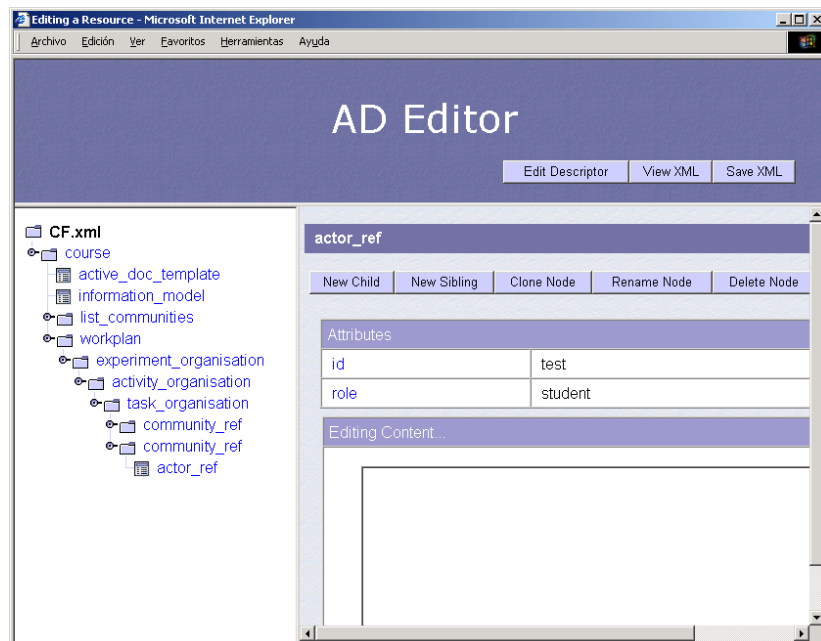


Figure 84. Required attributes for the *actor_ref*

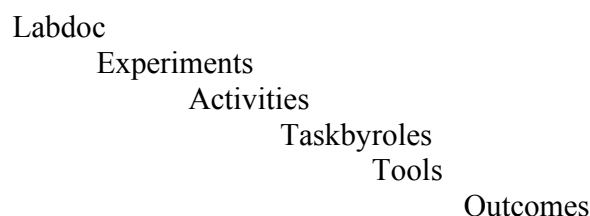
4.1.5 The results

The results files fulfil the called “AD_growable_v3.dtd”.

4.1.5.1 What is the results file?

The results file is the representation in XML of the objects built by the students in each of the tasks presented in an AD.

The basic structure it has is the following:



However this structure will vary depending on the AD that the results file belongs to.

An outcome represents a result stored by a DB tool: the type is defined in it, the date it was stored, the format, etc.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE user_active_doc SYSTEM "http://rigel.lsi.uned.es:4080/ADServer/Conf/AD_growable_v3.dtd ">
<user_active_doc user_id="test">
  <profile>User test</profile>
  <labdoc_ref id="labdoc01">
    <experiment_ref id="exp_1">
      <activity_ref id="activity_001">
        <taskbyrole_ref id="taskdef01">
          <tool_ref id="DrawTool">
            <outcome_ref date="13/6/2003 13:13:7" outcome_format="svg" outcome_type="text"
workarea_id="EditorMAref1"><![CDATA[<?xml version="1.0" standalone="no"?>
<!DOCTYPE svg><svg>
  <line x1="147.0" y1="44.0" x2="84.0" y2="120.0" style="stroke-width:1;stroke:rgb(255,0,0)"/>
  <line x1="84.0" y1="120.0" x2="184.0" y2="121.0" style="stroke-width:1;stroke:rgb(255,0,0)"/>
  <line x1="149.0" y1="43.0" x2="187.0" y2="119.0" style="stroke-width:1;stroke:rgb(255,0,0)"/>
</svg>]]></outcome_ref>
          </tool_ref>
          <tool_ref id="tool4">
            <outcome_ref date="13/6/2003 13:13:11" outcome_format="plaintext"
outcome_type="text" workarea_id="ref7"><![CDATA[Result saved]]></outcome_ref>
          </tool_ref>
        </taskbyrole_ref>
      <taskbyrole_ref id="taskdef01_m"/>
      <taskbyrole_ref id="taskdef02">
```

```

        <tool_ref id="tooltest">
        <outcome_ref date="13/6/2003 13:13:20" outcome_format="plaintext"
outcome_type="internal" workarea_id="ref6666"><![CDATA[a) Answer 1.]]></outcome_ref>
        </tool_ref>
        </taskbyrole_ref>
        <taskbyrole_ref id="taskdef03"/>
        <taskbyrole_ref id="taskdef03_m"/>
    </activity_ref>
</experiment_ref>
</labdoc_ref>
</user_active_doc>

```

When you enter the environment of an AD with the role of student, you find inside the “Navigation menu” an option that allows you to generate an XML results file associated with the student that has used the environment. (figure 89).

4.2 Working on an AD scenario as a student

To work in the system in the role of student, we open a navigator and connect to: http://localhost:catalina_port/ADServerSetup

In the section “Work AD” we select the AD that we want to work with (figure 85), and click on the button “Work as Student Role”.

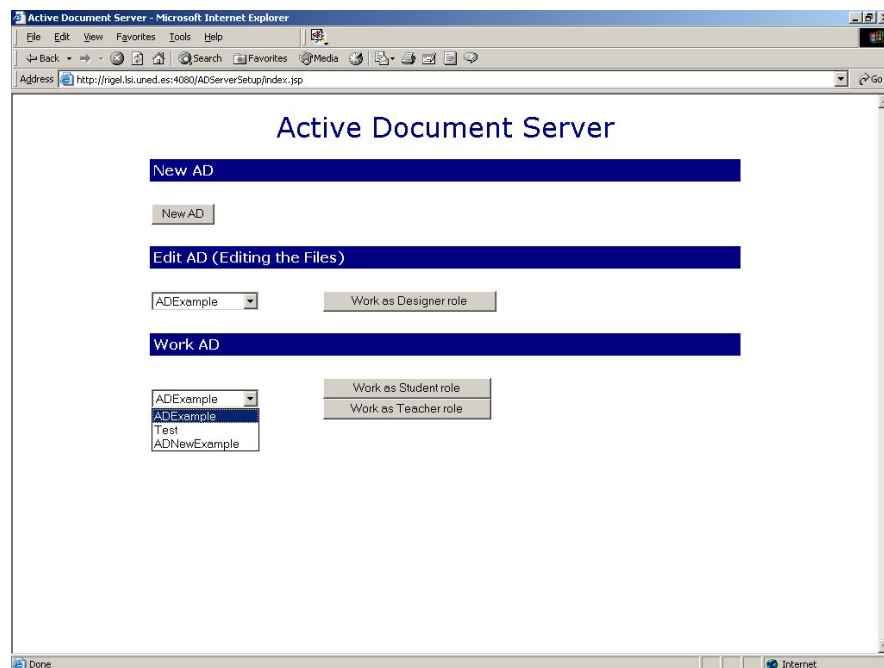


Figure 85. AD Server interface: selecting the AD

4.2.1 Login the system

The entrance point that gives access to the AD will appear (in this case “ADExample”) asking for the entry of the user’s access data (figure 86).

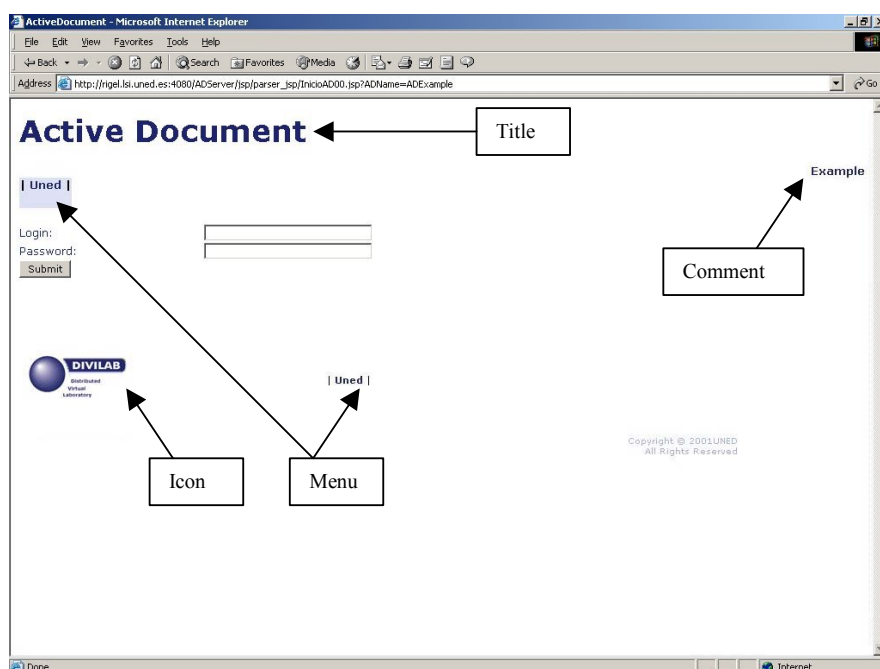


Figure 86. System access window

The users that can access the system are the ones that are registered in the table “User” belonging to the database used to store the results (in this case DBResults). By defect when creating an AD two users are created (one to work as student and the other as teacher) but if preferred, more can be added as it is described in point 4.1 *Working in the system as a learning scenario designer* in the section of *Databases*.

The opening page to the AD will ask for the inclusion of some data that is personal for each user: “login” and “password”, in this case we will use the user with the role of student created by defect upon entering the system (figure 87):

Login: test
Password: test

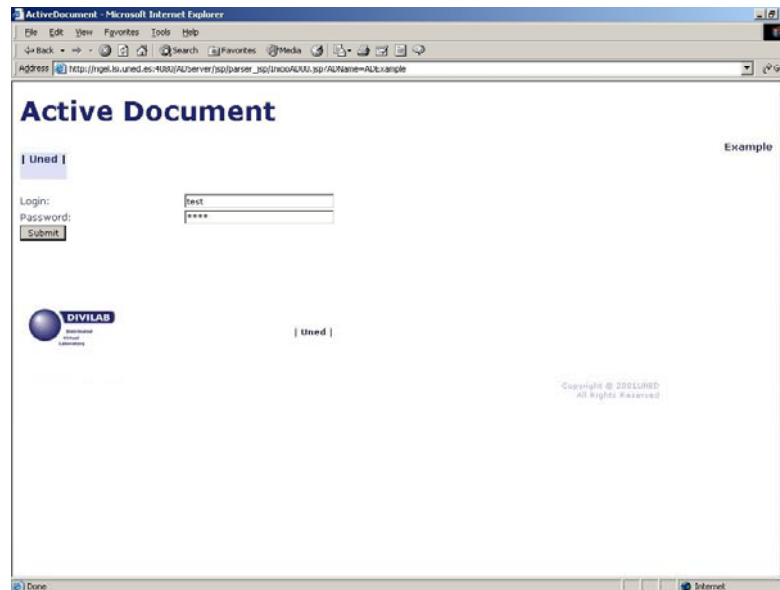


Figure 87. Login as *test* user

Once the solicited data has been entered, click on “Submit”, and this will make the ADServer validate whether the information has been entered correctly. If it hasn’t been, there will be a message indicating the error.

If the data is correct, two windows will appear (figure 88) that will ask if you wish to install and run the subprogram signed and distributed by “LTCS”, and you should answer “Si”. To avoid these windows appearing each time the environment is accessed, answer “Conceder siempre”. Note that if the Java run time has been installed in English the window will obviously present the options in that language and not in Spanish.



Figure 88. Installation dialog for the required software

4.2.2 The interface

Here is an environment that corresponds to our AD “ADExample” just as it is shown in figure 89.

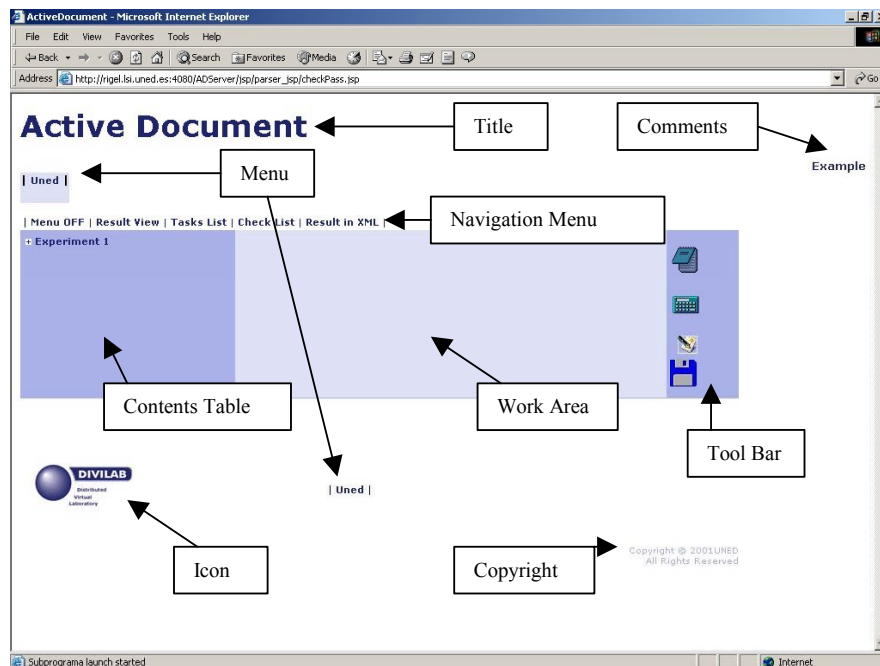


Figure 89. The Active Document interface

Depending on the configuration that has been selected, we can see the following elements (figure 89):

- Upper part:
 - At the very top, to the left you can see the title
 - On the left you can see the comment
 - Just below the title we can see one or two menu bars, if two appear, the first one is a menu that we have defined upon creating the AD.
 - The navigation menu (if there are two, the second menu) is from the application and it allows you to change the view of the AD. The available options are the following:
 - Menu OFF | ON: Allows taking out or showing the table of contents in the central left part.
 - Result View | Work View: If activated the view of the results that appear will be the results of the tasks the user has carried out. If the work view is activated in the AD you will see the tools that permit the user to carry out the tasks.

- Tasks List: Shows the list of tasks to be done along with the results of completed tasks
 - Check List: Teacher's comments about the tasks carried out by the student.
 - Result in XML: Shows the result of the tasks done by the student in XML format.
- Central left area:
Known as the contents table. It allows the user to move around the different sections that make up links to the experiments and defined activities in the AD, the link text is the title of the section that is being run. When you click on one of them the content unfolds in the central part of the window.
 - Central part:
Known as work area. Here the content of the section of the table of contents selected in each moment appears.
 - Central-right part:
Contains the auxiliary tool bars.
 - Lower part:
 - On the left side the icon of the AD appears
 - If a menu has been defined when the AD was created it will appear in the central part.
 - On the right side the copyright appears

4.2.3 How to interact with the environment

To see the activities defined within an experiment, all we have to do is to click on it. When we click on "Experiment 1" we can see the following changes (figure 90):

- In the table of contents, under the link of the experiment, the activities that the selected experiment has defined will appear.
- In the central part we see the description of the experiment.

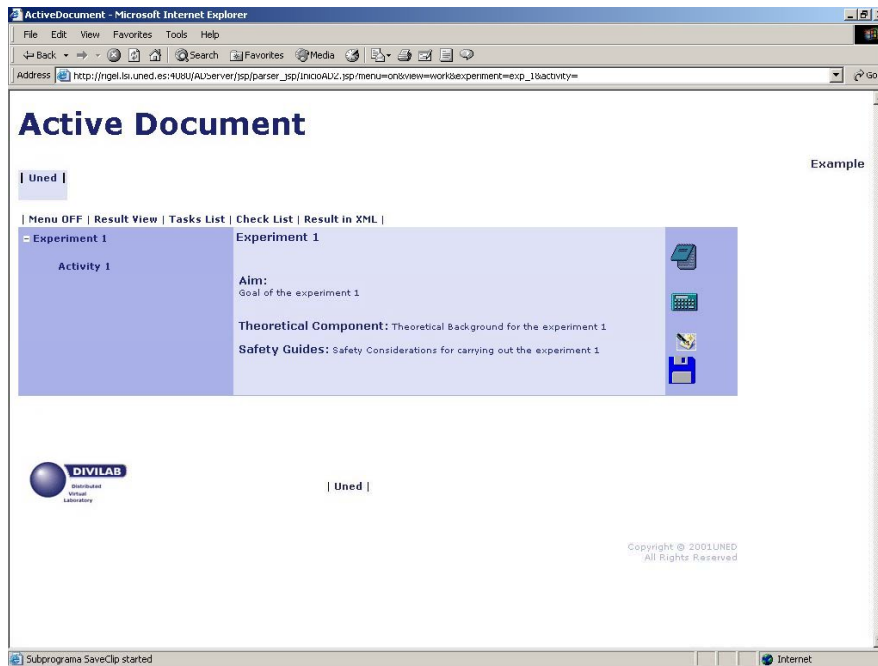


Figure 90. Working with the Active Document

When clicking on an activity we will see that in the workarea there is a description of the activity followed by the defined tasks that this AD has defined for the user that has accessed the environment.

For each task there will be a description and then, depending on the view (further documented on page 131) we are in, we will see:

- In the work view (figure 91): an example of the Draw Tool can be seen which has been associated with the task for the user to work with.

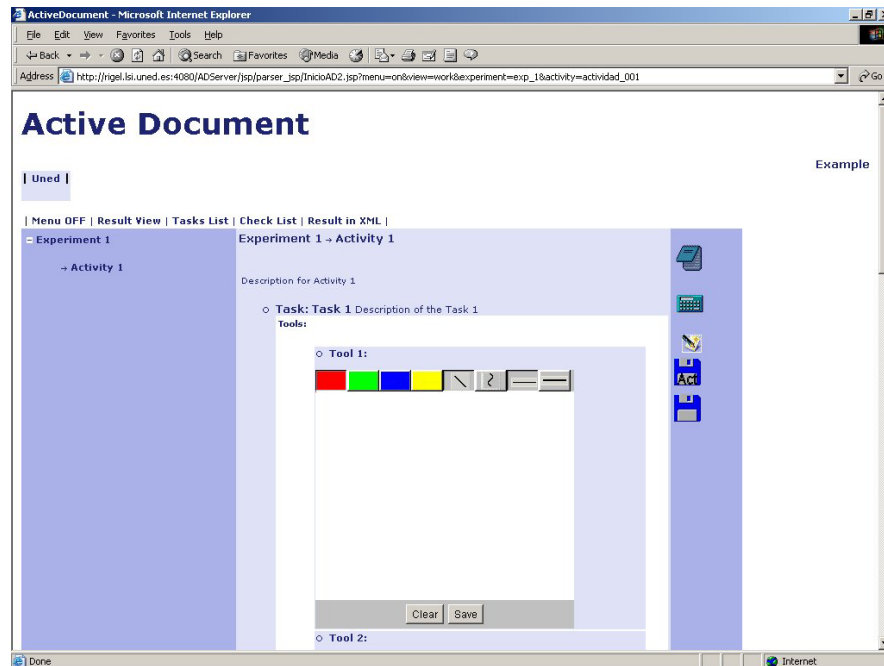


Figure 91. Using a tool for the *Experiment 1*'s *Task 1*

- In the results view (figure 92): the results the user has previously saved for this task can be seen, if something has been done.

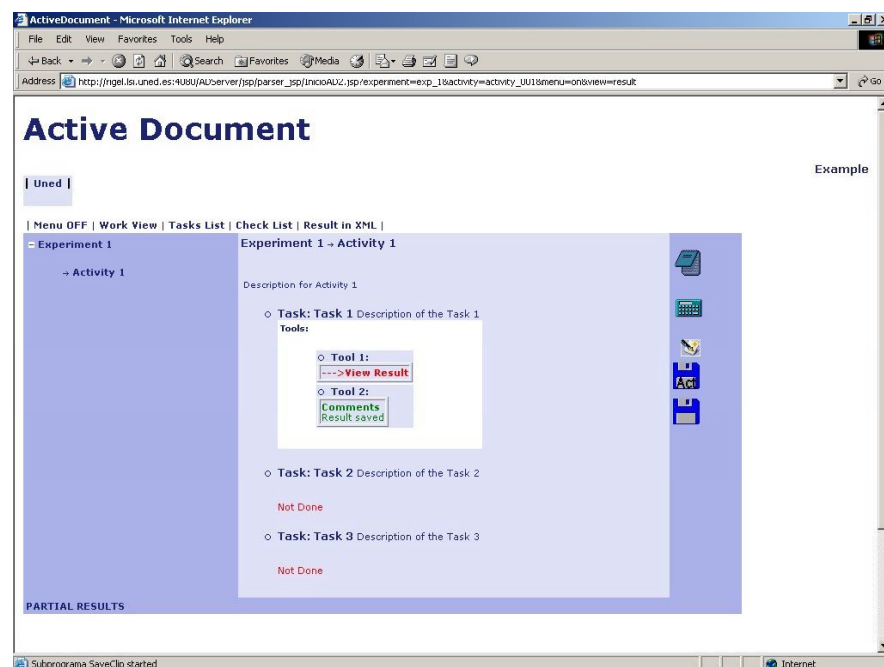


Figure 92. Results for the tasks which have been carried out

4.2.3.1 Use of internal tools within the tasks

Internat tools have been defined to be tools which only work with the AD system. Now we will show how internal tools appear within the AD environment:

- **editor**: allows you to do task that have written text answers (figure 93). When you click on “Save” it stores the answer on the results DB.

The image shows a software window titled "Comments". Inside the window is a large text area containing the text "Result saved". To the right of the text area is a vertical scrollbar. At the bottom of the window, there are two buttons: "Save" and "Reset".

Figure 93. The editor: adding comments

- **tooltest**: It allows you to carry out, answer and correct the tasks that have multiple-choice (figure 94). When you click on “Save” it saves the answer on the results DB.

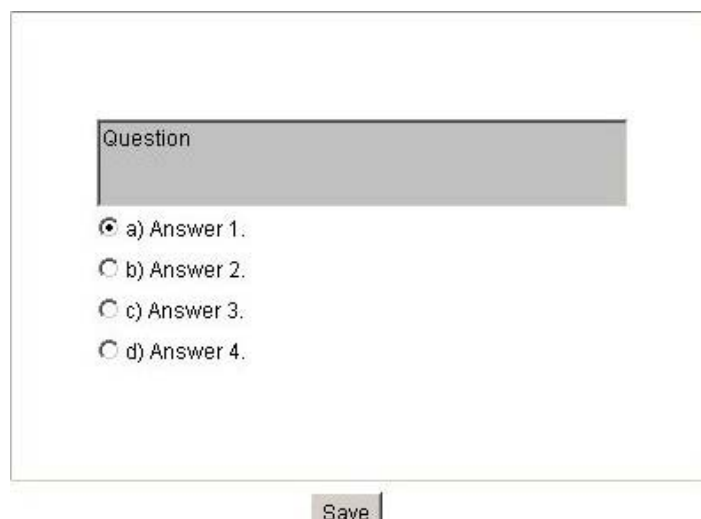
The image shows a software window for a multiple-choice test. At the top is a grey rectangular box labeled "Question". Below this box are four radio button options: "a) Answer 1.", "b) Answer 2.", "c) Answer 3.", and "d) Answer 4.". The first option, "a) Answer 1.", is selected. At the bottom of the window is a "Save" button.

Figure 94. Multiple-choice testing tool (**tooltest**)

- **DrawTool:** Allows you to carry out tasks that have answers in the form of drawings (figure 95). You choose a colour (it has four defined colours), one drawing mode (straight or curved) and a line thickness (thin or wide). When you click on “Save” it stores the answer on the results DB.

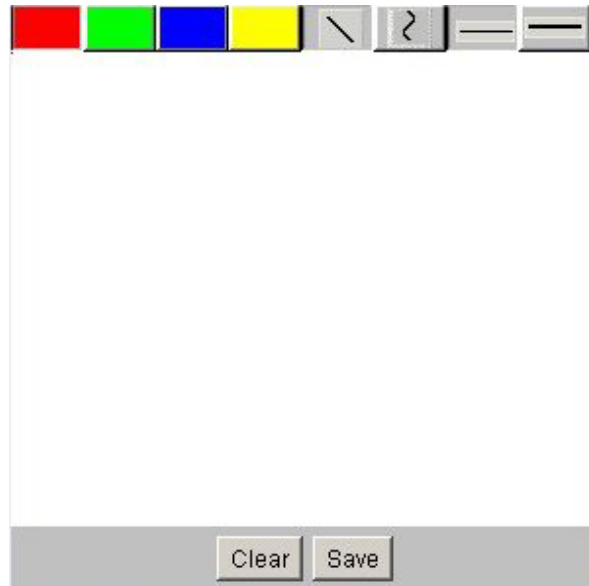


Figure 95. Drawing tool (*DrawTool*)

Now we can move on to consider how the navigation menu can be used.

4.2.3.2 Navigation menu

- **Menu off | on**
When you click on the link, it changes the view of the document (figure 96).

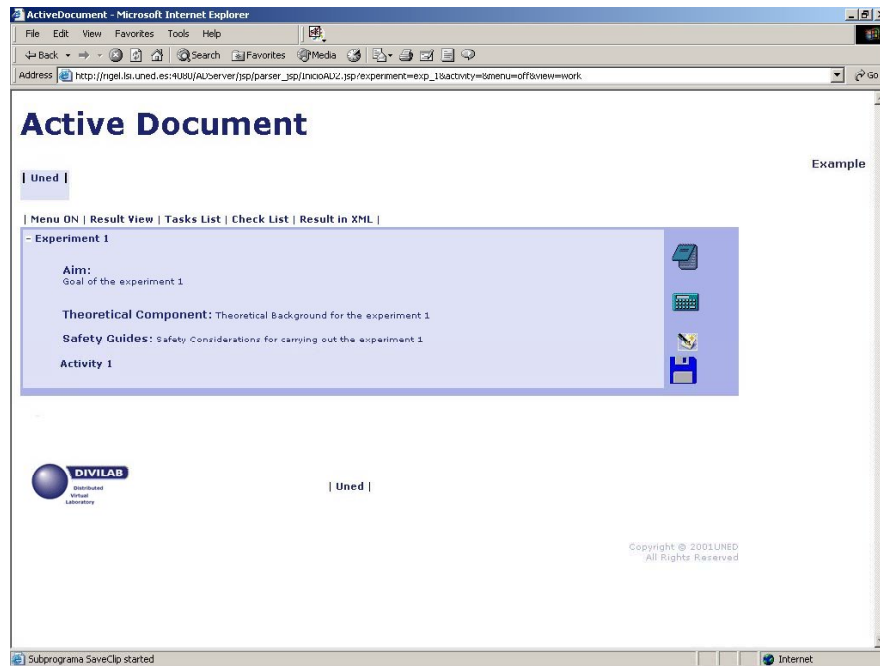


Figure 96. Changing the document view (menu off)

- ***Result View | WorkView***

Result View

This is a view of the AD that shows the AD with the results the user has produced for the tasks that were presented in the workarea of the project. With this view the user can only view the results, they cannot modify or add.

- ***Tasks List***

Allows the student to see the results produced for all the tasks defined in the document at the same time (figure 97).

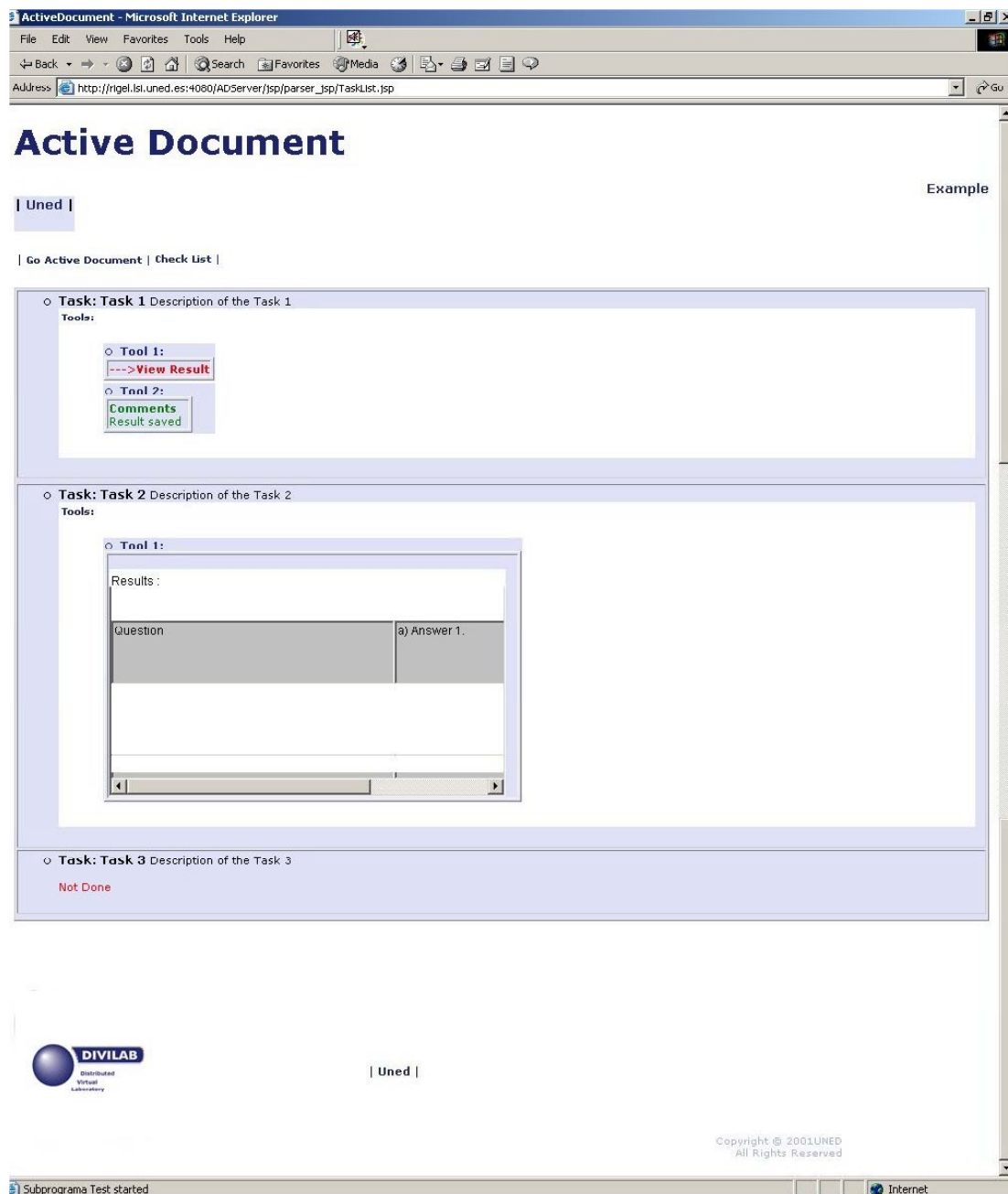


Figure 97. Displaying the state of the tasks

- **Check List**

Allows you to see the answers to all defined tasks that were given by the student along with the comments that the tutor might have made regarding them and the qualification obtained (figure 98). Obviously if no correction tool has been defined for this experiment then no comments can be made, so this would be empty.

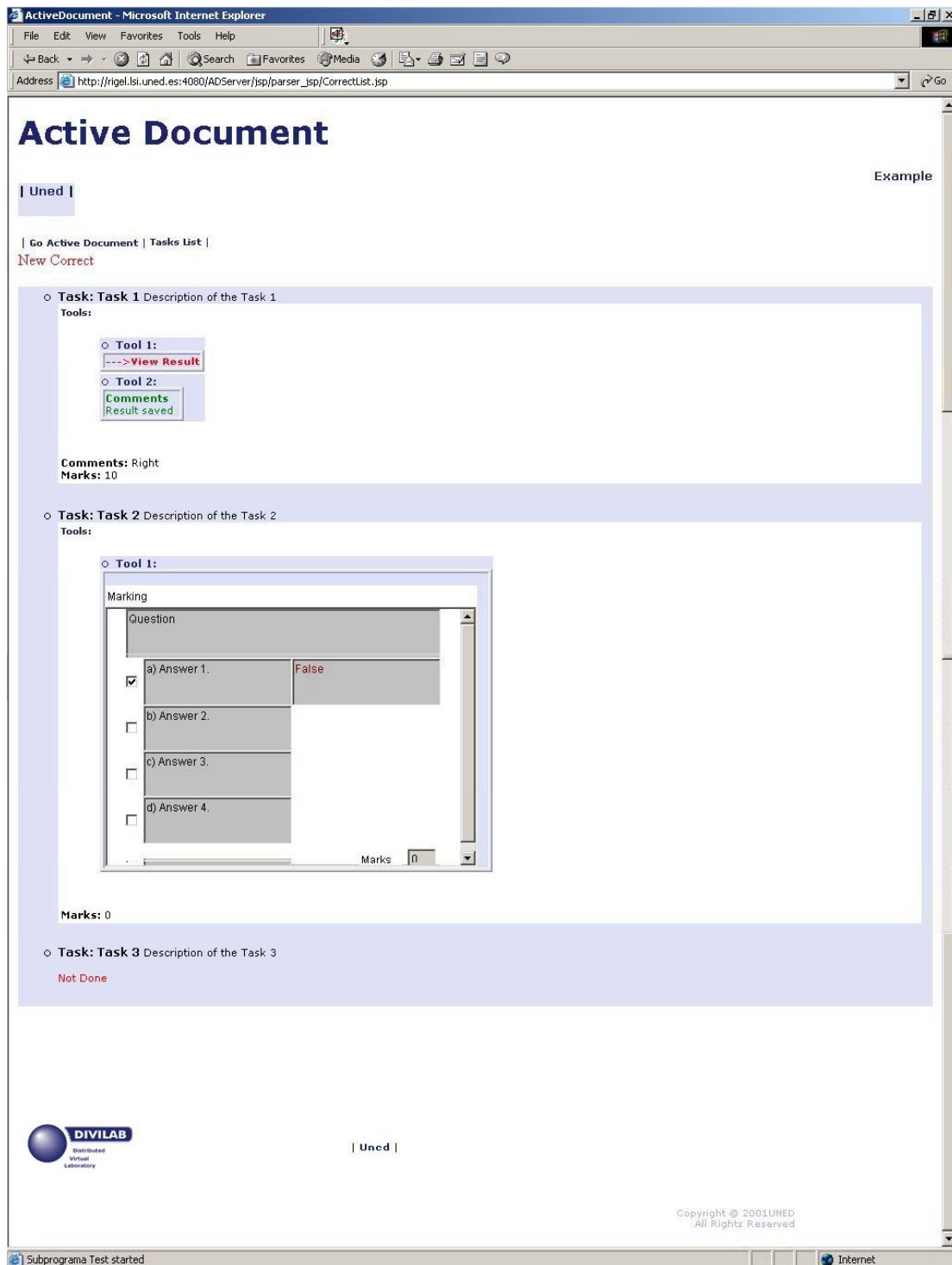


Figure 98. Viewing the student's answers

○ **Result in XML**

Allows you to see the answers given by the student to all the tasks in the XML format (figure 99).

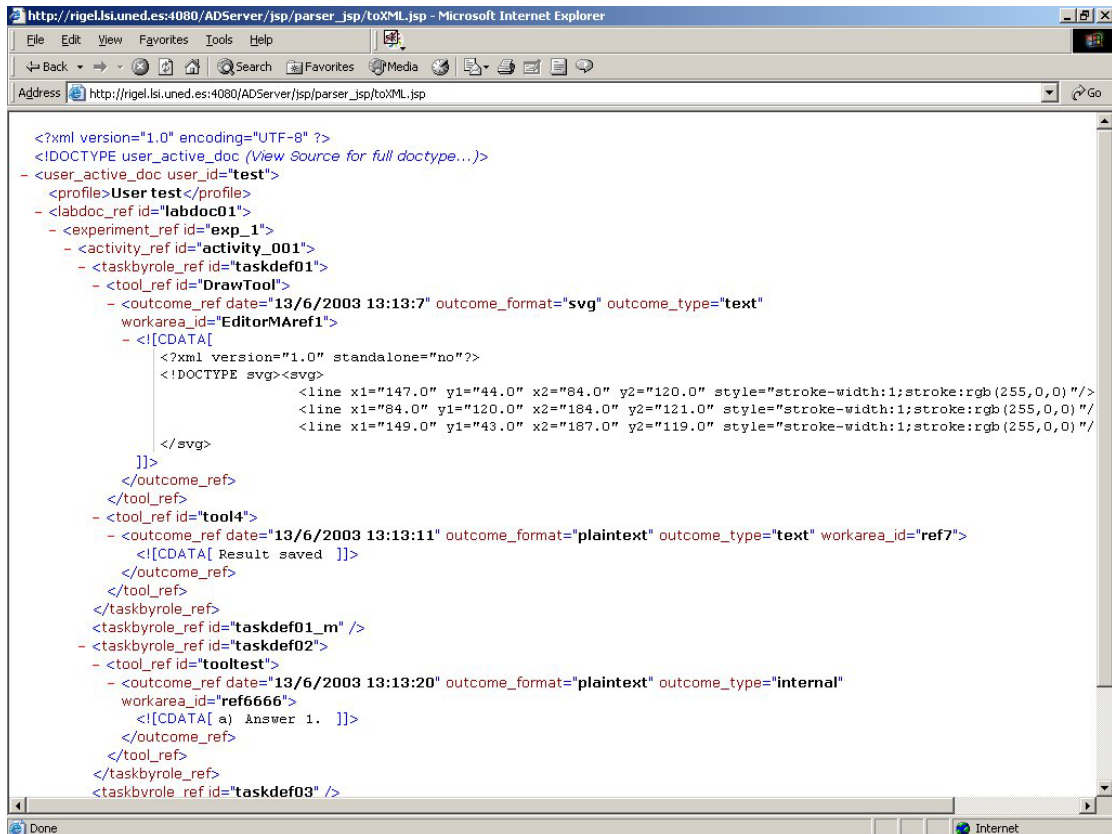


Figure 99. The student's answers in XML format

4.2.3.3 Tool Bar

In the right part of figure 100 we can find the toolBar, the exact composition of which would depend on the tools which have been defined. We will now describe its different components:

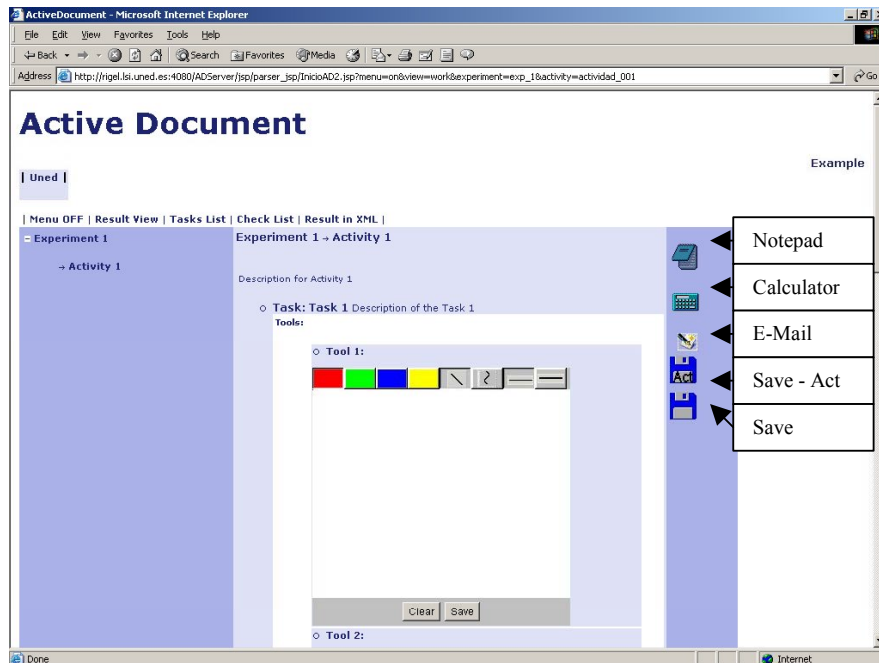


Figure 100. The toolbar's components

- **Notepad**
When you click on the icon, a Windows notepad is opened (this button only works with systems that use Windows).
- **Calculator**
When you click on the icon, the Windows calculator is opened (this button only works with systems that use Windows).
- **Email**
When you click on this button the message editor defined in the system is opened.
- **Save associated to the AD**
When you click on this button, the contents of the clipboard are saved as a partial result associated to the AD.
- **Save associated to the activity**
When you click here the contents of the clipboard are saved as a partial result associated to the task of the AD that is active at that time.

4.2.3.4 Prerequisites revisited

If in the AD prerequisites have been defined, the student will not be able to see the part that is restricted until the imposed conditions have been fulfilled (figure 101), if they haven't been fulfilled, a message will appear indicating what condition still has to be met.

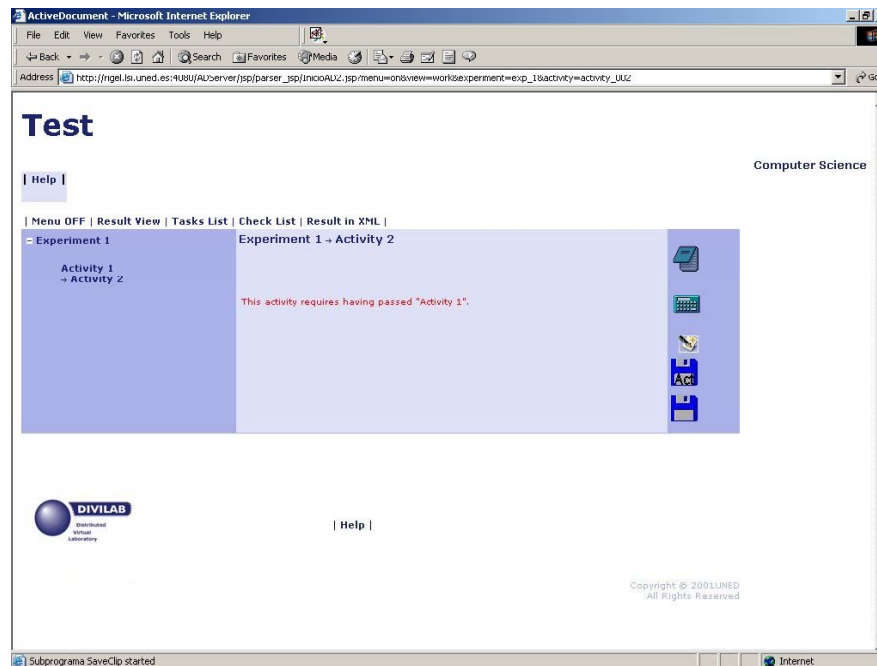


Figure 101. An error message: a prerequisite has not been satisfied

If the AD has defined prerequisites of the type “done”, the state of visualization of the AD for the student will vary depending on the results entered.

4.2.4 Errors that can be produced when using the system in the role of student

- **Icons are missing in the toolbar:** The user hasn't given permission to the signed applet or does not have j2re. Give permission to the applet as explained in point 4.2 *Working on an AD scenario as a student* in the section *Login the System*. Check the installation of the j2re.
- **Connection time is over. Please Reconnect:** When the system is inactive for a specific amount of time, the message from figure 102 indicates to the user that the connection time has run out. Connect again with the ADServerSetup and select the Active Document again.



Figure 102. Timed out connection message

- **Cannot connect to MySQL server**: There are connection problems with the database server (figure 103), the most probable cause is that it is down. Run the DB server.

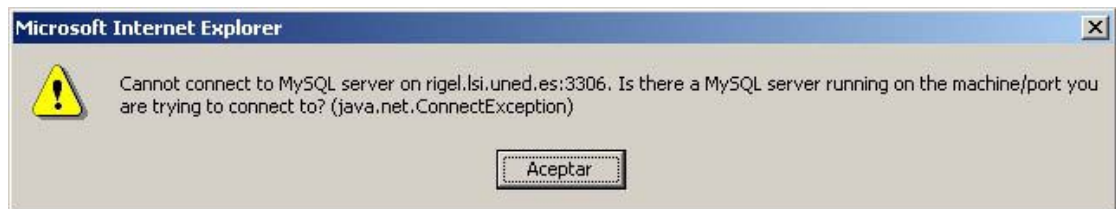


Figure 103. Unable to connect to MySQL server dialog

- **Warning! An unexpected error**: When the error from figure 104 appears, and no warning window comes up, we have to look at the Tomcat console (figure 105) to see what the real error is. Normally it is due to some of the XML files that the AD has do not fulfil the DTD. If the XML files do not fulfil the DTD, the environment will not work.

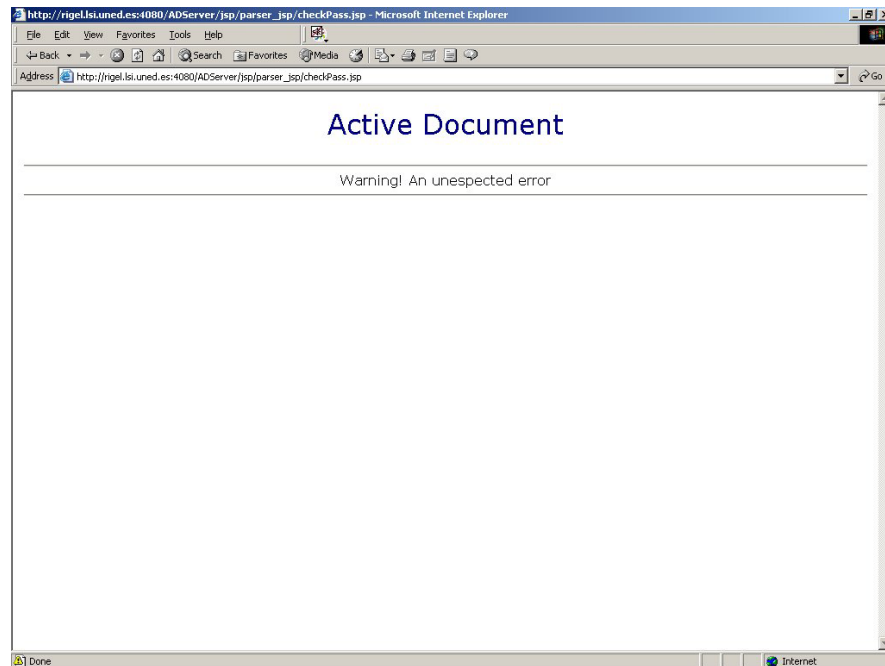


Figure 104. Reporting an unexpected error

```

C:\> Acceso directo a tomcat-bin - catalina run
idOutcome: outSaveClipGlo
time of creation: 1058440774669

ResultSession with sessionId: c01;test;global;outSaveClipGlo already in hashtable
Configuration File: c:\tomcat\webapps\ADServer\Conf\Configuration1.0.xml
ADName: ADEExample
AD File: c:\tomcat\webapps\ADServer\Conf\ADEExample\AD.xml
Resources File: c:\tomcat\webapps\ADServer\Conf\ADEExample\RL.xml
Community File: c:\tomcat\webapps\ADServer\Conf\ADEExample\CF.xml
DBControl v1.1
-----Query is: SELECT passwd FROM users where login='test' AND passwd='test';
test
-----Query is: SELECT passwd FROM users where login='test' AND passwd='test';
test
-----Query is: select usernum from users where login='test';
[Error] AD.xml:22(linea):54(columna):: Element type "fff" must be declared.
Error c:\tomcat\webapps\ADServer\Conf\ADEExample\AD.xml :Element type "fff" must
be declared.
** New ADProcessor Error: org.xml.sax.SAXParseException: Element type "fff" must
be declared.

```

Figure 105. Tomcat console: session output

If it is a problem with one of the XML files, edit the file so that it fulfils the DTD.

- **Visualization errors appear:** When visualization errors appear when showing defined tasks in an activity or when showing a list of tasks, it may be because tool identifiers that are not defined in the RL.xml are used within the file AD.xml.
- **Associated tasks do not appear when they should:** Go over the community file, the user who entered into the system might not have any tasks associated to that activity.

4.3 Working on an AD scenario as a teacher

4.3.1 Login the system

To work in the system in the role of teacher, we open the navigator and we connect to: http://localhost:catalina_port/ADServerSetup

In the section “Work AD” we select the AD that we want to work with and we click on the button “Work as Teacher Role” (figure 106).

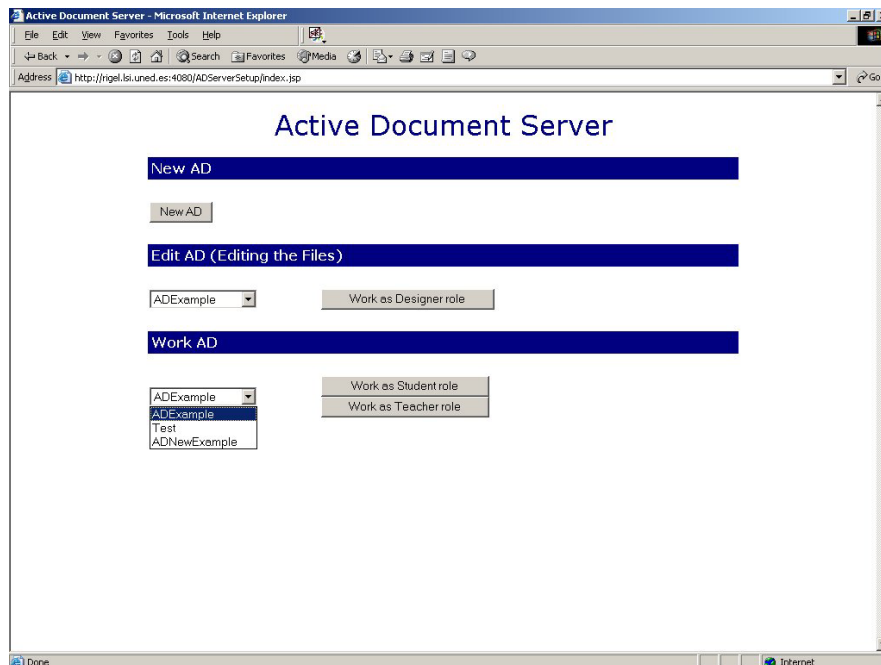


Figure 106. Choosing an AD to work as a teacher

A tutor or teacher has two different options when interacting with the system. Monitor the work that the students are carrying out, see their progress, etc., or actually correct the work, as can be seen in figure 109. A window will appear that gives access to the correction document in the Active Document chosen, asking for the user access data to be entered (figure 107).

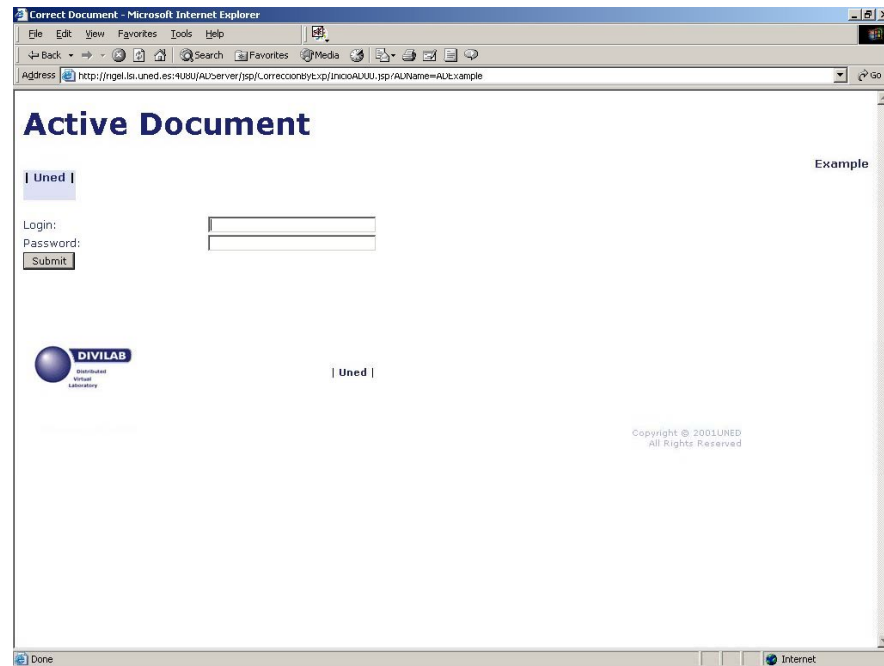


Figure 107. System access window

The opening page to the AD will ask for the inclusion of some personal data for each user: “login” and “password”, in this case we will use the user with the role of teacher created by defect to access the system (figure 108):

Login: **marker**
Password: **marker**

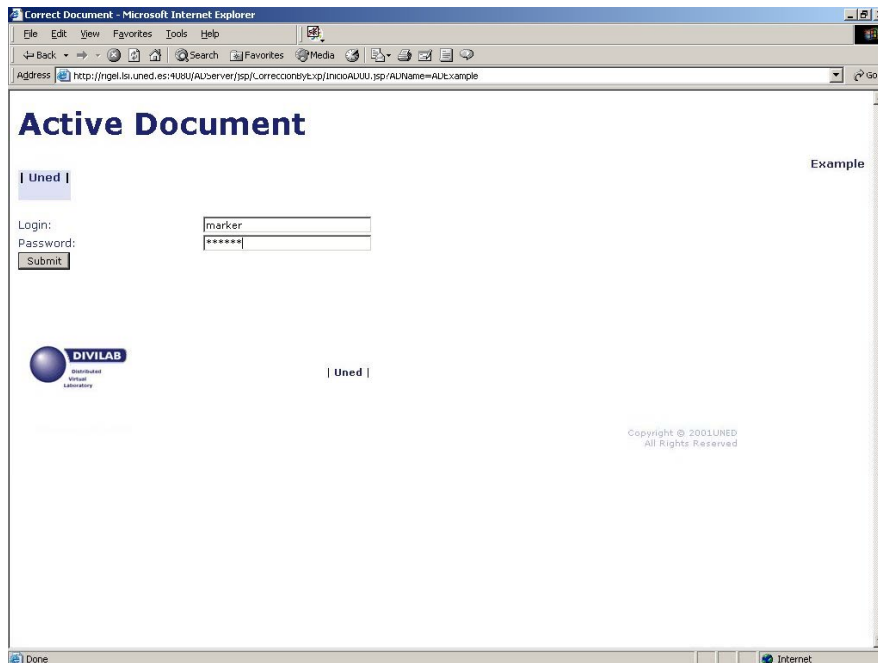


Figure 108. Login as teacher (*marker*)

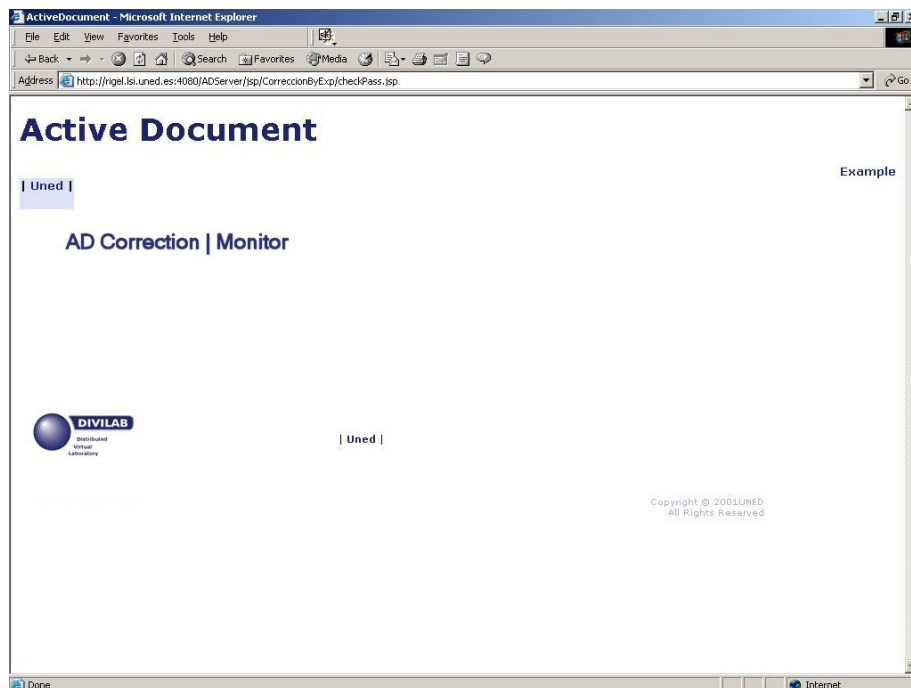


Figure 109. AD Correction or Monitor Selection

4.3.2 The Monitor

This tool has been developed by the UDUI research group, and as such, makes use of other tools they have previously built.

Program functionality – a brief technical description.

The basement of the monitor is built by the CoolModes-JavaProgram. It is a (shared) workspace environment that provides a dynamic palette handling. Each palette can be used to design a workspace as needed with graph based layouts. We chose this program because of its flexible structure and because the TM/AD structure itself makes it obvious to display themselves as graphs.

The monitor has a basic palette which consists of three elements

1. a general information of the current position in the navigation tree.
2. a node specific information panel that describes valuable information like participants taking part, global and participant related status and describing elements that are able to interact with the tutor (e.g. E-Mail address of a participant is displayed and can be clicked to open a dialogue that sends him an e-Mail.).

This information is displayed when the tutor touches a node (e.g. an experiment or task) with the mouse or selects it. If he selects it the information will be persistent even if the mouse leaves the component.

3. a legend that describes the important elements the monitor consists of. For example, there are the colours that represent a specific state (like running, finished, aborted), the icons and their meaning/functionality, the basic node itself as it appears in the workspace and possible prerequisite types (represented by different edges in the graph) an item can have.

Furthermore there will be palettes with filter components, that can be simply dropped into a workspace, to be able to sift the contents of a TM request as well as a palette that allows the tutor to define alarm “clocks” which inform him when specific conditions appear (like a user misses a given date to deliver his work).

The workspace itself will display a closed graph consisting of one “Top Node” which represents the highest item in the hierarchy level that is chosen and its directly descending sub items (e.g. As TopNode we have an experiment that shows its activities). These sub items are connected by “prerequisite”-edges in the manner they depend on themselves.

To navigate the tutor simply clicks an icon at a corresponding node. It is possible to let a sub item show its sub items, where all other nodes disappear to have a clean view, it is possible to step higher in the hierarchy and show the TopNode along with its next higher item and all of this nodes sub items, it is possible to show a directly preceding or succeeding item (relative to the current TopNode) and its sub items, it is possible to show the participants of an item or how they are organised in communities and at last it

is possible to show all of the above with the focus on a specific participant.

The program structure in detail is firstly based on the palette design of CoolModes. Each displayable Node is cut in three parts, a Model holding the data, a View responsible for displaying the data to the user and a controller handling events and interaction with the user and between model and view. These nodes are held in a separate Java-package in the palettes tree of CoolModes. In the palettes package itself there is a class for every palette that has to be displayed in the application.

How is the connection to the TaskManager is done?

The TaskManager provides a http-based interface for sending queries to it, where it is possible to add various predefined parameters to the URL of the http connection in order to specify the type and the depth of information requested. The output delivered to the client by the http server is an XML document conforming to the TaskManager DTD. This XML document basically contains information on a course, its participants and the status of their activities. The depth of information can be adjusted by a parameter of the TaskManager URL. Depending on the depth the status of the experiments, the activities belonging to each of the experiments of the course, and of tasks that are part of an activity. It contains as well the prerequisites for each of the experiments, activities or tasks.

Cool Modes receives this XML document as an input stream. It is then parsed by a DOM XML parser, and a structure of Java objects is created that is identical to the tree structure of the XML input stream.

To get all information necessary for visualization in Cool Modes, it is sometimes necessary to perform multiple http requests to the TaskManager server. From each of these requests, such a structure of Java objects is created. The information gathered from these object structures is then combined to create the nodes in Cool Modes, that represent a course, an experiment, an activity or a task, and the edges between the nodes representing the prerequisite relationships between them. The structure of the nodes and edges is quite similar to the original object structure as created by parsing the Task Manager XML stream. Only some Cool Modes-specific information must be added, for instance each node must have a unique ID.

The UI - Information overview and navigation

The UI can be best explained by showing some examples. A course named “Course IT2”, which are illustrated in figures 110 – 114.

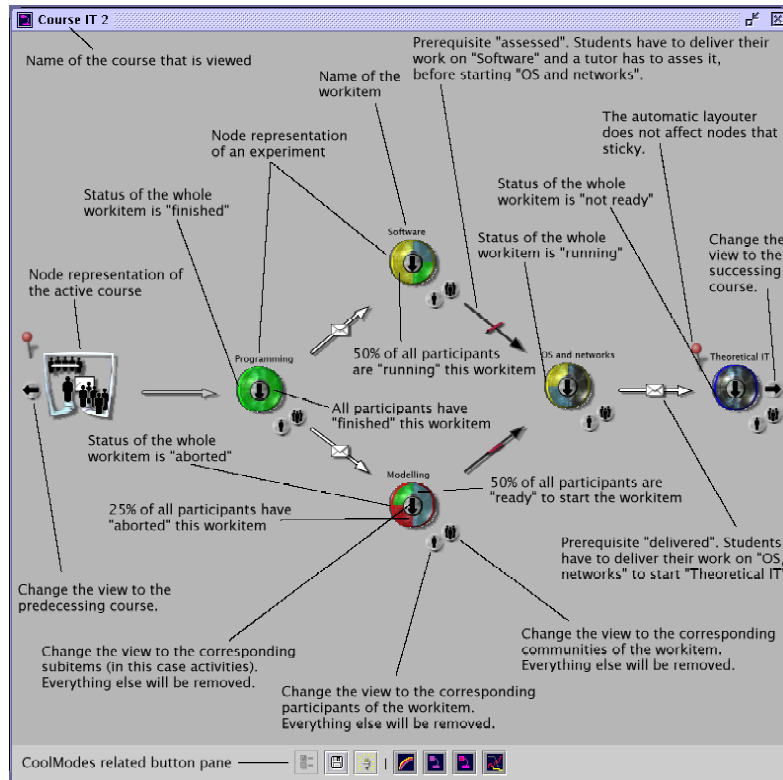


Figure 110. Aspects of the Monitor interface

Succeeding we have the experiment Software:

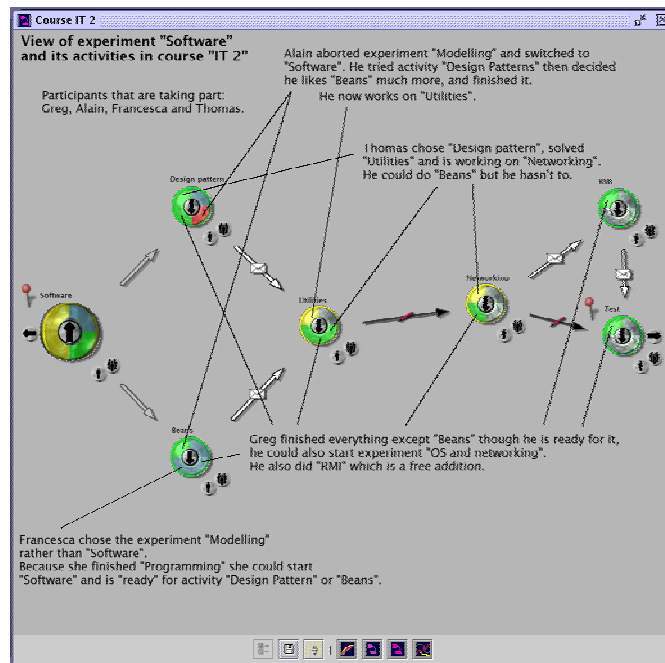


Figure 111. Aspects of the Monitor interface

Of course the participants can be viewed (here at activity “Design pattern”):

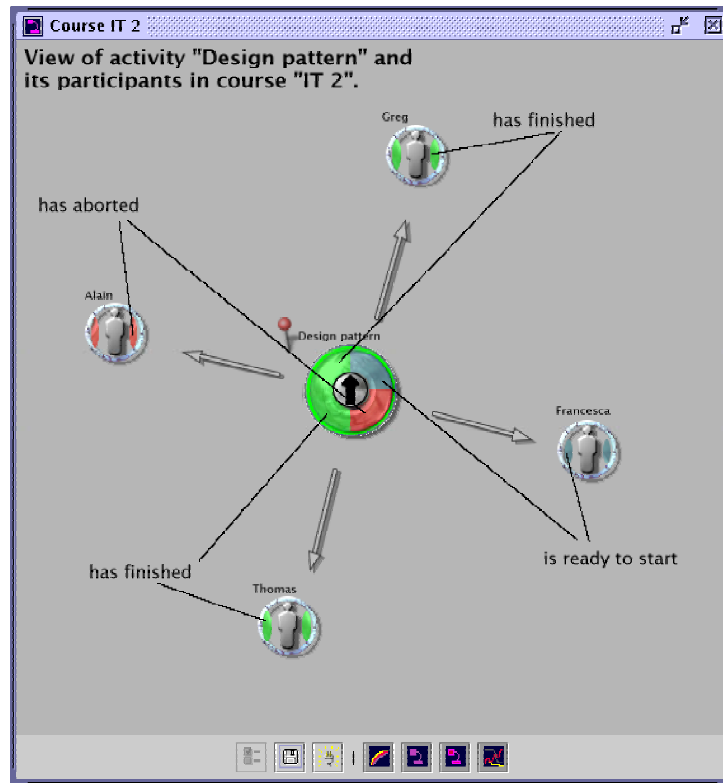


Figure 112. Aspects of the Monitor interface

To the right of the workspace we find the palette:

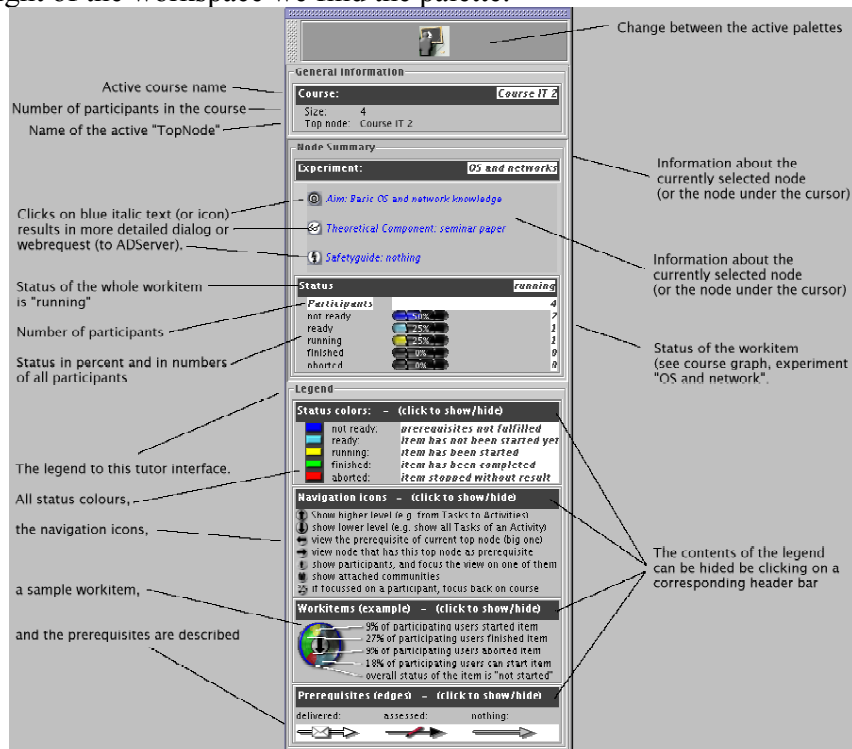


Figure 113. Aspects of the Monitor interface

The legend can be shortened:

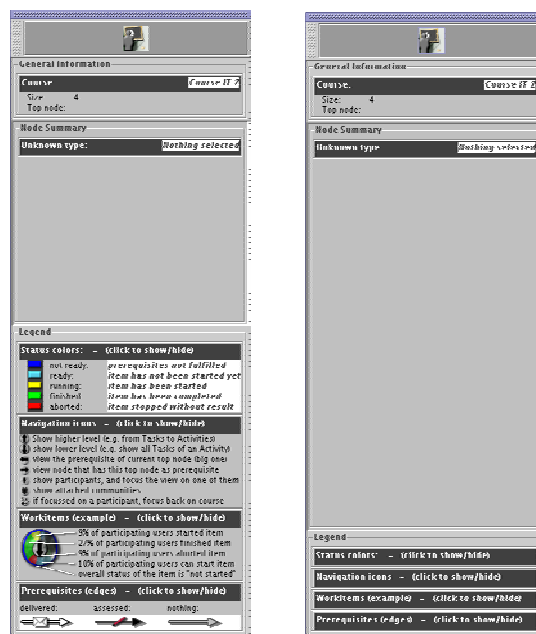


Figure 114. Aspects of the Monitor interface

4.3.3 The AD Corrector

Once the user data is valid, a window appears that shows the communities that are on the AD and within each one of them it shows the students that belong to each community. (figure 115).

Depending on the distribution of element within the page, we can see the following elements:

- Upper part:
 - At the very top, on the left you can see the title
 - On the right you can see the comment
 - If a menu has been defined when the AD was created, we can see it just below the title.
 - Name of the teacher that accessed the environment.
- Central part:
 - If the user that has entered has got assigned correction tasks within this AD, they appear grouped by community, the students that have assignments to do on this AD, they have been assigned a correction tool.
- Lower part:
 - In the left part you can see the AD icon
 - If a menu has been defined when the AD was created, it will appear in the central part.
 - On the right you can see the copyright

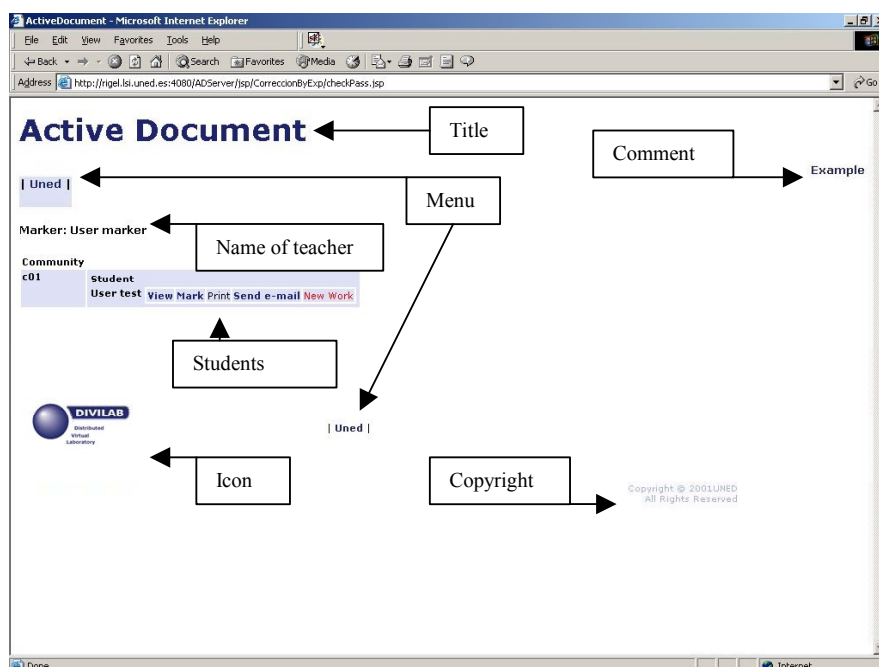


Figure 115. The Active Document's teacher's interface

Next to the community identifier, you can see the names and surnames of the students in this community that have tasks to be corrected.

Next to the names and surnames of each student you can see a series of links and a message that indicates the state it is in:

- **View:** It allows you to see the qualifications obtained by the student in the experiments and activities defined on the AD
- **Mark:** Allows the correction of the AD
- **Print:** Not used at the moment
- **Send e-mail:** Allows you to send an electronic mail message to the address that is stored in the database for that student.
- **Status:** Indicates what stage the student is in with the carrying out of the AD. It can have the following values:
 - *Not Started:* Has not begun yet
 - *Nothing New:* There is nothing new
 - *New Work:* New tasks are available

4.3.4 How to interact with the correction tool

4.3.4.1 Marking

When you click on the link “*Mark*” associated with a student, a window will appear that shows a list of experiments defined in the AD along with this student’s mark for that experiment (figure 116). If the student still has not been marked, no qualification will appear.

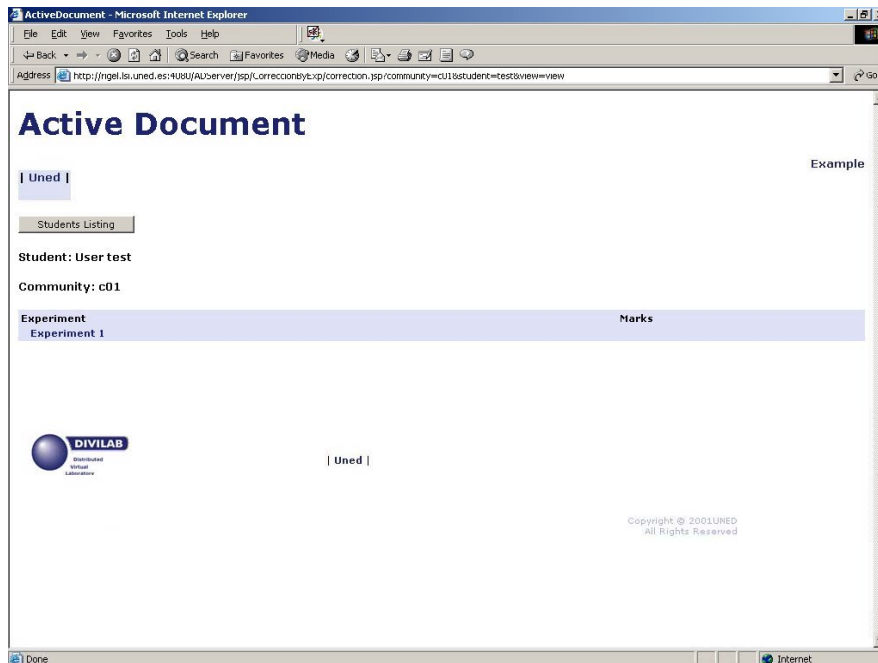


Figure 116. Accessing the students' answers

When you click on an experiment, you will see the defined activities that contain tasks where the teacher appears as “marker” in the community file, along with the answer that the student has given (figure 117).

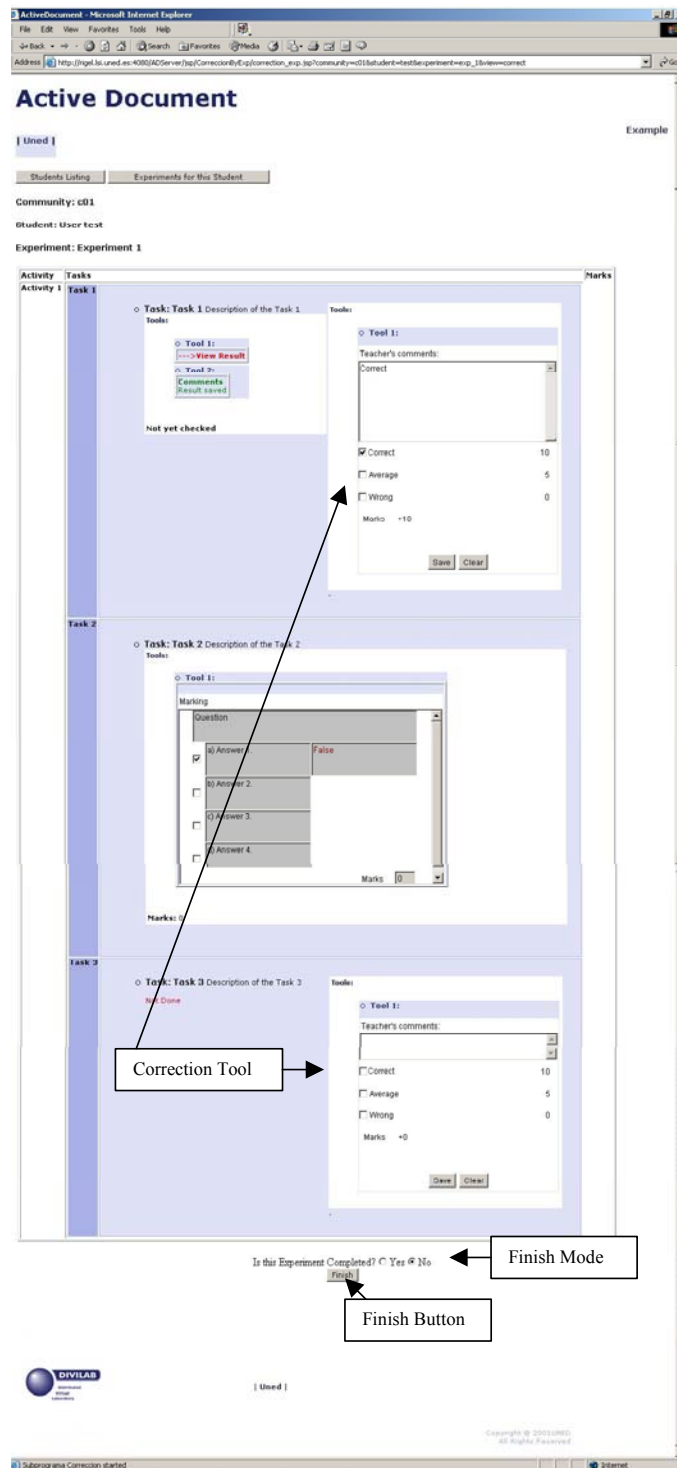


Figure 117. Marking the student's answers

If in the AD.xml a task has been associated a correction, and in that correction has a defined correction tool, you will see a correction tool along with the student's answer which will allow the tutor to qualify the task.

Once the tutor finishes correcting the results generated by the student in experiment, the student should find out if the experiment is finished or not. To do this, select the appropriate value of Finish mode and then click on the "Finish Button".

If the value of Finish mode is "no" (figure 118), the tutor considers that the experiment is still not finished, and cannot calculate the average mark of the activities nor the experiment.

It is like this because in some cases it is not necessary to finish all the activities that make up an experiment in order to carry it out, the tutor must consider whether the finished activities are appropriate to fulfil the experiment.

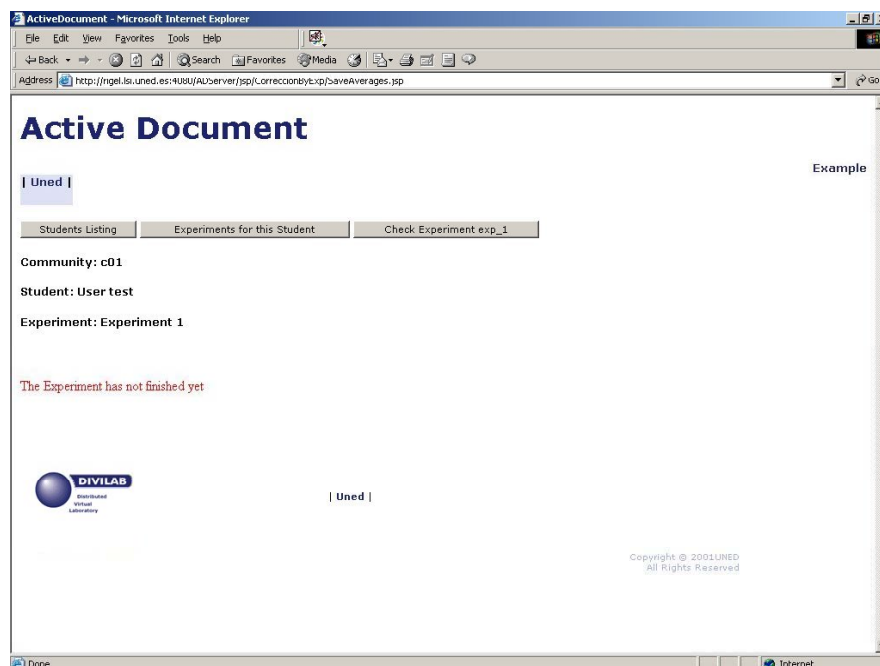


Figure 118. The current experiment has not finished yet

If the value of Finish mode is "yes" (figure 119), the tutor considers that the experiment is finished and calculates the average mark of the experiment and the activities that it is made up of.

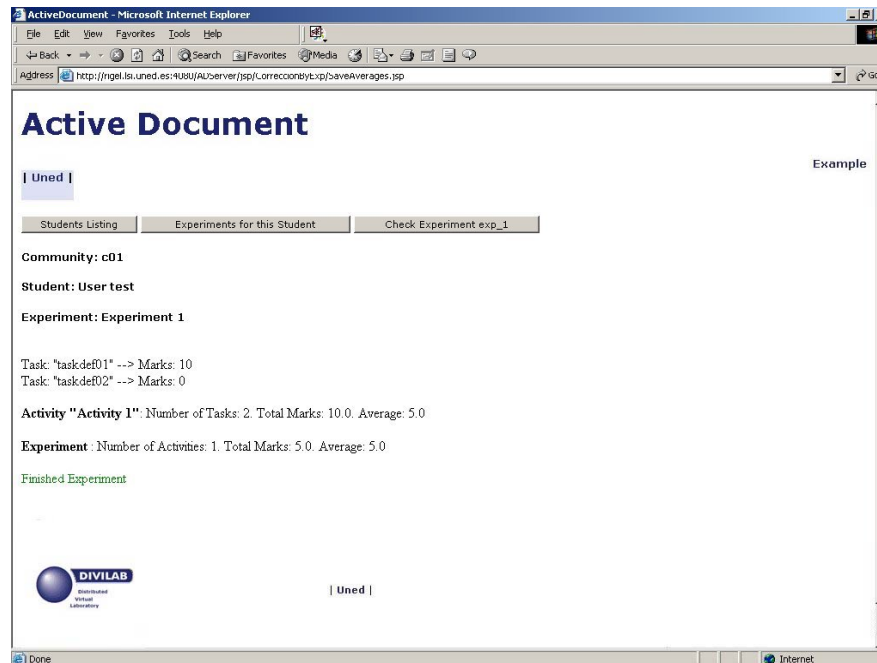


Figure 119. A finished experiment

The buttons that appear in the upper part allow:

- Students Listing: allows you to go back to the page where the student list appears.
- Experiments for this Student: allows you to go back to the page where the experiments that are related to this specific student appear.
- Check Experiment "...": allows you to go back to the page where the experiment appears "." that are related to this specific student.

4.3.4.2 Viewing

When you click on the link "View" associated with a student, you can see a window that shows a list of experiments associated to the AD (figure 120).

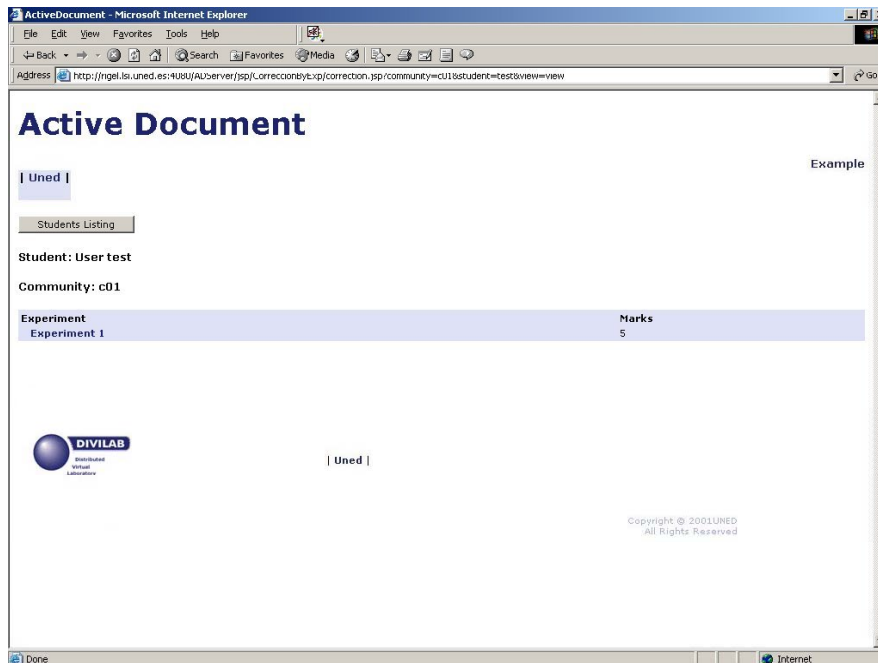


Figure 120. Viewing the experiments associated with an Active Document

When you click on an experiment you can see the activities defined in the experiment that have tasks where the teacher appears as “marker” in the community file, along with this student’s mark for this task (figure 121).

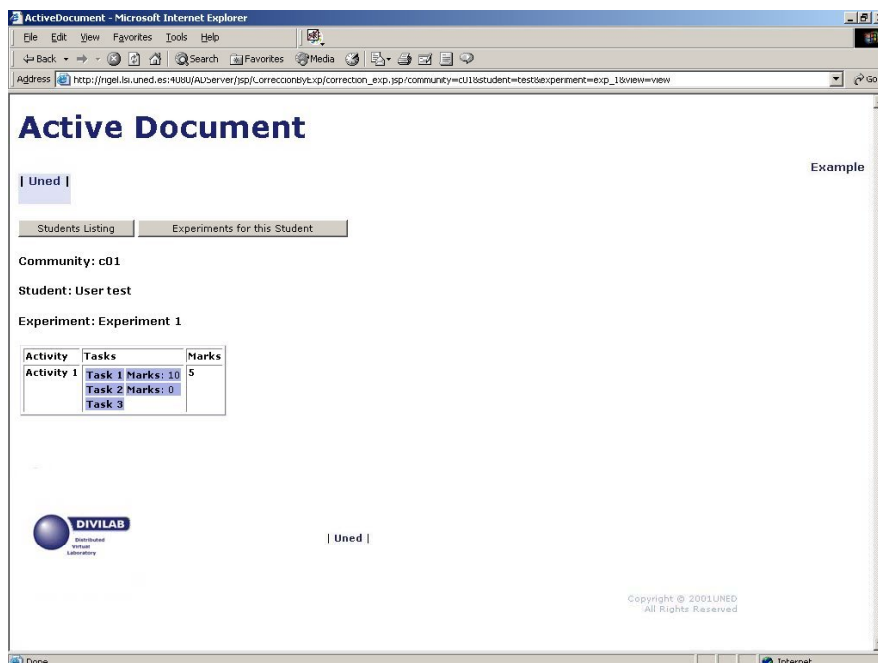


Figure 121. Activities of the current experiment that the teacher has to mark

The buttons that appear in the upper part allow:

- Students Listing: Allows you to go back to the page where the list of students appears.
- Experiments for this Student: Allows you to go back to the page where this specific student's experiments appear.

4.3.4.3 Details of the correction tool

This tool (figure 122) allows the teacher to mark and comment on a task done by the student, and when you click on the button “Save” it stores the comments and marks in the results database.

Teacher's comments:

Correct

<input checked="" type="checkbox"/> Correct	10
<input type="checkbox"/> Average	5
<input type="checkbox"/> Wrong	0

Marks +10

Save Clear

Figure 122. Marking a task and adding comments for the student

If the AD has been defined with prerequisites like “correct” or “passed” the student's visualization state of the AD will vary depending on the corrections and marks entered by that student's tutor.

4.3.5 Errors that can happen when using the system with the role of teacher

Apart from the errors indicated in point 4.2 *Working on an AD scenario as a student* within the section *Errors that can happen when using the system with the role of student* you might see the following:

4.4 Eliminating an AD

To eliminate an AD follow these steps:

- Eliminate the results and log databases that are being used in the AD.
- Eliminate the directory that can be found in the ADServer that corresponds to the AD.
- Edit the configuration folder to eliminate the part which corresponds to the AD.
- Stop and run Apache again.

5 Application

A brief example application which illustrates some of the interesting aspects of the AD system is included in deliverable D7.5, and a more detailed tutorial will be provided on the AD System web site <http://sensei.lsi.uned.es/ActiveDocument>

Index of Terms

Active Document, 6, 7, 9, 13, 26, 28,
36, 37, 47, 53, 82, 93, 105, 107, 123,
125, 134, 137, 145, 151
Active_doc_template, 53, 105
ActiveDocument, 5
ActiveDocument Parameter, 39
Activity, 54
Activity_organisation, 105
Actor, 105
Actor_ref, 106
AD, 5, 19, 20, 21, 22, 23, 24, 25, 26, 27,
28, 29, 30, 39, 43, 45, 46, 47, 49, 50,
52, 53, 63, 64, 65, 75, 76, 93, 96, 104,
109, 119, 120, 121, 123, 124, 125,
127, 129, 134, 135, 136, 137, 138,
146, 150, 152, 153
AD client, 33
AD system, 20, 21, 23, 25, 27, 46
AD-definition document, 53
ADServer, 9, 10, 11, 12, 13, 14, 15, 16,
17, 18, 19, 30, 31
ADServerSetup, 7, 8, 9, 16, 18, 31
Aim, 54, 56
architecture, 5
Associated to tasks, 53
attributes, 48, 67, 68, 69, 70, 71, 73, 74,
75, 79, 80, 86, 99, 100, 101, 102, 103,
111, 116, 117, 118
Bd_object, 54
Bd_relation, 55
BinaryPathO, 20, 22
BinaryPathR, 20, 22
Calculator, 133
CatalinaHome, 20, 21, 28
CatalinaPort, 20, 21, 28
Check List, 130
child nodes, 48, 66, 98
Comment, 20, 26, 37
community, 110, 113
Community, 106
Community Definition, 45
community XML file, 109
Community_ref, 106

Configuration, 10, 12, 13, 19, 20, 35,
36, 37, 39
ConfigurationAD, 20, 21
ConfigurationSystem, 20, 21
Content Information, 48, 101
Content_ref, 55, 56
Context Name, 13, 17, 18, 30
ContextName, 21, 23
correction, 28, 29, 53, 59, 61, 89, 90,
91, 137, 145, 146, 152
correction tool, 89, 149
Course, 107
Database, 12, 19, 35, 37
DataBaseLog, 21
DatabaseObject, 21, 22
DatabaseResult, 21, 22
databases, 12, 13, 15, 17, 19, 31, 36,
153
date_in, 58, 60, 75, 76, 79
date_out, 58, 60, 75, 76, 79
DBLog, 15, 17, 19, 21, 24, 31, 37
DBResults, 15, 19, 22, 25, 31, 36, 121
defining a scenario, 52
Description AD, 45, 46, 93, 104
DirResult, 22, 27
Document, 93
DrawTool, 29, 39, 53, 83, 96, 119, 128
DriverO, 22
DriverR, 22
DTDs, 44, 49
Edit AD, 47
Editing, 47, 51, 64, 96, 109
editor, 48, 49, 50, 53, 64, 70, 72, 82, 86,
92, 96, 100, 109, 112, 127, 133
Email, 133
errors, 16, 18, 44
Errors, 19, 44, 134, 152
Experiment, 56, 58, 69, 76, 124, 150
Experiment_organisation, 107
File, 93
General purpose, 53
GeneralConf, 20, 22, 23
Glossary, 56

hardware and software requirements, 6
 Host Name, 11
 Hostname, 21, 23, 27
 Icon, 23, 26, 38
 Identifier, 39
idTask, 89
 Information_model, 107
Inserting, 65
 installation, 7, 8, 9, 12, 15, 16, 18, 19, 31, 34
 interact, 146
 interact with the environment, 124
 interface, 123, 145
 internal too, 127
 internal tools, 82
 Java Home Path, 12
 JavaHome, 21, 23, 27
 JdbcDriver, 23
 labdoc, 65
 Labdoc, 57, 119
Linux, 7, 8
 List_prerequisites, 58
 List_communities, 107
 List_roles, 58
 Log, 21, 23, 24, 26, 30
 Logging, 37
 Login, 17, 18, 137
 LogXML, 24
 Mark, 146
 marker, 89
 markerTask, 89
 Mediating_tools, 58
 Menu, 24, 26, 123
Menu off| on, 128
 MenuOption, 24, 25, 26
 Metadata, 59
 MetadataEditor, 15, 17, 18, 19, 31, 47, 64, 96, 109
 mySQL, 12
 MySQL, 7, 9, 12, 13, 19, 23, 25, 35, 36, 43, 46, 135
 MysqlPort, 25
 Name, 18, 24, 25, 38, 55, 56, 57, 94, 95, 105
 NameO, 22, 25
 NameR, 22, 25
 Navigation menu, 128
new activity, 70
 new AD, 19, 34, 47, 52
 New Attribute, 68, 70, 73, 77, 85, 88, 91, 99, 111, 115, 117
new task, 72
Notepad, 133
 Object, 59
 OS, 21, 25, 27
outcome Definition, 46
 Param, 59
 ParamConf, 25, 26, 28
 Parameter, 60, 94
 Parameter Tool, 39
 PasswdO, 22, 26
 PasswdR, 22, 26
 Password, 13, 17, 18, 26, 36, 43, 121, 138
 PathLog, 24, 26
 Port Number, 12
 Prerequisite, 60
 prerequisites, 75
 Prerequisites, 134
 Presentation, 21, 23, 26, 27, 37, 39
 Reference, 60
 Resource, 94
resource definition, 46, 94
 resource XML file, 96
 resource_ref, 84
 Resource_ref, 61
Result in XML, 131
 Result View, 129
 results, 119
 ResultXML, 21, 27
 Role, 62, 94, 120, 137
 role of a student, 43
 role of a teacher, 43
 role of student, 134
 safety_guides, 56, 62, 66
 Safety_guides, 62
Save associated to the activity, 133
Save associated to the AD, 133
 See, 21, 48, 54, 56, 58, 59, 60, 62, 63, 94, 95, 96, 106, 107, 108

- Server, 7, 9, 12, 13
- ServerO, 22, 27
- ServerR, 22, 27
- student, 120
- Style, 26, 27
- Stylesheet, 38
- System, 7, 21, 27, 33, 121, 134, 138
- task**, 113
- Task_organisation, 108
- taskbyrole, 72
- Taskbyrole, 62
- Tasks List*, 129
- teacher, 137
- Theoretical_component, 63
- theoretical_components, 66
- Tip, 63
- Tip_content, 63
- title, 24, 28, 44, 56, 58, 59, 68, 70, 71, 72, 76, 95, 123, 124
- Title, 26, 28, 37
- Tomcat, 5, 7, 8, 9, 12, 15, 16, 17, 18, 20, 21, 28, 30, 46, 135, 136
- Tool, 95
- tool bar, 48
- Tool Bar, 132
- ToolConf, 28, 29
- Tools, 21, 28, 29, 39, 46, 53, 119
- tooltest**, 29, 53, 83, 120, 127
- Un-installation*, 16, 18, 19, 30
- Updating after the changes, 17
- URI_ADServlet, 23, 29
- URI_BaseApplet, 23, 29
- URI_Community, 19, 29
- URI_RL, 19, 20, 29, 30
- UriDTDLog, 24, 30
- UriDTDResult, 27, 30
- URL, 23, 27, 28, 39, 44
- User, 12, 13, 30, 36, 119, 121
- UserO, 22, 30
- UserR, 22, 26, 30
- Web Server Home Path, 12
- Windows**, 7, 33, 34
- Workarea, 96
- Workplan, 108
- Xerces, 8
- XML, 5, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 37, 44, 45, 46, 48, 49, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 93, 94, 95, 96, 104, 105, 106, 107, 108, 119, 120, 124, 132, 135, 136

Table of Figures

Figure 1. The DiViLab prototype	6
Figure 2. Decompressing the installation file	8
Figure 3. Accessing the ADServer installation	10
Figure 4. Accessing the ADServer installation setup	11
Figure 5. (Setting up) Configuring the ADServer installation on our systems	14
Figure 6. Error during the ADServer installation	15
Figure 7. Tomcat console	16
Figure 8. ADServer installation finished succesfully	17
Figure 9. Java plugin installation.....	34
Figure 10. Creating a new AD.....	35
Figure 11. Configuring a new AD.....	41
Figure 12. Error creating a new AD	42
Figure 13. New AD created succesfully	43
Figure 14. Accessing the ADServer	47
Figure 15. Choosing the AD to edit.....	48
Figure 16. Selecting the AD File.....	50
Figure 17. Adding roles to a new role list (<i>list-roles</i>)	50
Figure 18. Editing the role list identifier (<i>id</i>)	51
Figure 19. Editing a new role (student).....	52
Figure 20. Editing the AD Files.....	64
Figure 21. Selecting a <i>labdoc</i> node	65
Figure 22. creating a new experiment node.....	66
Figure 23. Structure for a new experiment	66
Figure 24. Filling in the content for the aim attribute	67
Figure 25. The content information attributes for an experiment.....	67
Figure 26. Adding the attribute <i>name</i> to a new experiment	68
Figure 27. Giving the name to an experiment	69
Figure 28. Experiment structure: attributes and content information.....	69
Figure 29. Adding a new activity to an experiment	70
Figure 30. Creating the attribute <i>name</i> for a new activity	71
Figure 31. Filling in the title of a new activity	71
Figure 32. A new activity. Attributes <i>name</i> and <i>title</i> and their values	72
Figure 33. Creating a new <i>taskbyrole</i> node for an activity.....	73
Figure 34. Adding an <i>identifier</i> to a new <i>taskbyrole</i>	73
Figure 35. Adding a list of possible roles for a task to a <i>taskbyrole</i> element.....	74
Figure 36. Inspecting the attributes of a new <i>taskbyrole</i> element	75
Figure 37. Adding a new list of prerequisites (<i>list_prerequisites</i>) to an experiment.....	77
Figure 38. The <i>list_prerequisites</i> element needs an attribute <i>id</i> (identifier).....	77
Figure 39. Filling in the identifier (<i>id</i> attribute) for the <i>list_prerequisites</i> element	78
Figure 40. Adding a new prerequisite (<i>prerequisite</i> node).....	79
Figure 41. Attributes for defining a new prerequisite	79
Figure 42. A new prerequisite and its attributes.....	80

Figure 43. Adding a new prerequisite for the current experiment.....	81
Figure 44. Identifying the experiment's prerequisite.....	81
Figure 45. Attributes (including a <i>prerequisite</i>) and content for an <i>experiment</i>	82
Figure 46. Adding <i>mediating_tools</i> to a <i>taskbyrole</i>	84
Figure 47. Adding a <i>resource_ref</i> to a <i>mediating_tools</i> node	85
Figure 48. Adding an identifier (<i>id</i>) to a <i>resource_ref</i>	85
Figure 49. Filling in the <i>id</i> 's value for the <i>resource_ref</i>	86
Figure 50. Attributes for a <i>resource_ref</i>	86
Figure 51. Adding a new <i>parameter</i> (a list of <i>param</i>) to a <i>resource_ref</i>	87
Figure 52. Adding a new <i>param</i> (an actual parameter) to a <i>parameter</i> (a <i>param</i> list).....	88
Figure 53. Adding the attribute <i>name</i> to a new <i>param</i>	88
Figure 54. Naming the <i>param</i> : this parameter's name will be 'text'	89
Figure 55. Attributes for a <i>taskbyrole</i>	91
Figure 56. Adding a <i>markerTask</i> to a <i>taskbyrole</i>	92
Figure 57. Filling in the <i>markerTask</i> 's name for the current <i>taskbyrole</i>	92
Figure 58. Selecting the resource file (RL.xml)	97
Figure 59. Default resource file with the basic tools included	97
Figure 60. Adding a new <i>resource</i> to the <i>resource_list</i>	98
Figure 61. Adding <i>metadata</i> and a <i>tool</i> to a new <i>resource</i>	99
Figure 62. Adding the attribute <i>id</i> (identifier) to a new <i>resource</i>	100
Figure 63. Filling in the name of the resource (<i>id</i>)	100
Figure 64. Attributes and content information for a new <i>resource</i>	101
Figure 65. Adding the (standard) <i>metadata</i> for a new <i>resource</i>	102
Figure 66. Inspecting the <i>tool</i> 's attributes	102
Figure 67. A <i>tool</i> requires a <i>workarea</i> and a <i>parameter</i> (a list of <i>params</i>).....	103
Figure 68. Attributes for defining a <i>workarea</i>	103
Figure 69. A <i>param</i> (an actual parameter named 'temperature')	104
Figure 70. Choosing the community file (CF.xml)	109
Figure 71. <i>course</i> structure showing the relevant nodes <i>list_communities</i> and <i>workplan</i>	110
Figure 72. Adding a new <i>community</i> node to a <i>list_communities</i>	111
Figure 73. Adding the attribute <i>id</i> (identifier) to a <i>community</i>	111
Figure 74. Filling in the value for the <i>community</i> 's <i>id</i>	112
Figure 75. Adding a new <i>actor</i> to a <i>community</i>	112
Figure 76. Adding the attribute <i>id</i> for the new <i>actor</i>	113
Figure 77. A <i>community_ref</i> including an <i>actor_ref</i>	114
Figure 78. Assigning a community (by means of <i>community_ref</i>) to a <i>task</i>	115
Figure 79. Adding the attribute <i>id</i> to a <i>community_ref</i>	115
Figure 80. Filling in the community's <i>id</i> value for the <i>community_ref</i>	116
Figure 81. Adding the actor who will be responsible for this community.	117
Figure 82. Creating the attribute <i>id</i> for the new <i>actor</i>	117
Figure 83. Filling in the value of the <i>actor_ref</i> 's <i>id</i> attribute	118
Figure 84. Required attributes for the <i>actor_ref</i>	118
Figure 85. AD Server interface: selecting the AD.....	120
Figure 86. System access window	121

Figure 87. Login as <i>test</i> user.....	122
Figure 88. Installation dialog for the required software	122
Figure 89. The Active Document interface	123
Figure 90. Working with the Active Document	125
Figure 91. Using a tool for the <i>Experiment1's Task1</i>	126
Figure 92. Results for the tasks which have been carried out	126
Figure 93. The editor: adding comments.....	127
Figure 94. Multiple-choice testing tool (tooltest)	127
Figure 95. Drawing tool (<i>DrawTool</i>)	128
Figure 96. Changing the document view (menu off)	129
Figure 97. Displaying the state of the tasks.....	130
Figure 98. Viewing the student's answers.....	131
Figure 99. The student's answers in XML format	132
Figure 100. The toolbar's components	133
Figure 101. An error message: a prerequisite has not been satisfied	134
Figure 102. Timed out connection message	135
Figure 103. Unable to connect to MySQL server dialog.....	135
Figure 104. Reporting an unexpected error	136
Figure 105. Tomcat console: session output	136
Figure 106. Choosing an AD to work as a teacher	137
Figure 107. System access window	138
Figure 108. Login as teacher (<i>marker</i>)	139
Figure 109. AD Correction or Monitor Selection	139
Figure 110. Aspects of the Monitor interface.....	142
Figure 111. Aspects of the Monitor interface.....	142
Figure 112. Aspects of the Monitor interface.....	143
Figure 113. Aspects of the Monitor interface.....	144
Figure 114. Aspects of the Monitor interface.....	144
Figure 115. The Active Document's teacher's interface.....	145
Figure 116. Accessing the students' answers	147
Figure 117. Marking the student's answers.....	148
Figure 118. The current experiment has not finished yet	149
Figure 119. A finished experiment.....	150
Figure 120. Viewing the experiments associated with an Active Document.....	151
Figure 121. Activities of the current experiment that the teacher has to mark.....	151
Figure 122. Marking a task and adding comments for the student	152

