COLDEX

Collaborative Learning and Distributed Experimentation

Information Society Technologies Programme

Project number: IST-2001-32327

Coldex System Architecture Summary

Deliverable Number: D.6.3.1
Contractual Date of Delivery: M24
Delivery Date: July, 2004
Version: 1.0 (Final)
Lead Partner: UNED
Contributing partners: UNED
Authors: Verdejo, M.F.; Barros, B.; Read, T.; Mayorga, J.I:, Velez, J., Calero, Y, Celorrio, C. & Lorenzo, E.J.
Contact: bbarros@lsi.uned.es

Revision history

Date	Name	Version	Description	Authors
September 2003	Prototype 0	0	No Distribution First version of the Ontology in KAON	UNED Group
May 2004	Prototype 1	1.0	Second version of the Ontology in Protegé Distribution with LDAP	UNED Group
July 2004	Prototype 2	2.0	Second version of the Ontology in Protegé Distribution with Hibernate Search options with the ontology The social model	UNED Group

General Architecture Functionality

Each Coldex System installation is an architecture which is oriented as a service provider based upon a three layer structure:

1. **The view layer**: is responsible for presenting data and offering users a way to interact with the system via the Web. It takes the form of an access portal.
2. **The model layer**: handles the execution of all the services that makes up the system, and is made up of the following sub-systems: the Knowledge Manager. (KM), the Social Manager (SM) and the Repository Service (LOR).
3. **The persistence layer**: offers the mechanisms necessary to provide persistence for all the storage needs of the system. It is made up of the following elements: the database, the ontology and the synchronization manager.

The architecture of this system can be seen in figure 1 below. It has been the result of several design cycles, and had to satisfy several important requirements, namely, it had to be distributed, extensible, keep semantic information (collected in a declarative way, for increased ease of use), adaptable and, as much as possible, close to the standards. As can be seen in the figure, the portal accesses the system services via a set of actions, which also access data in the persistence layer, necessary to store and retrieve information.

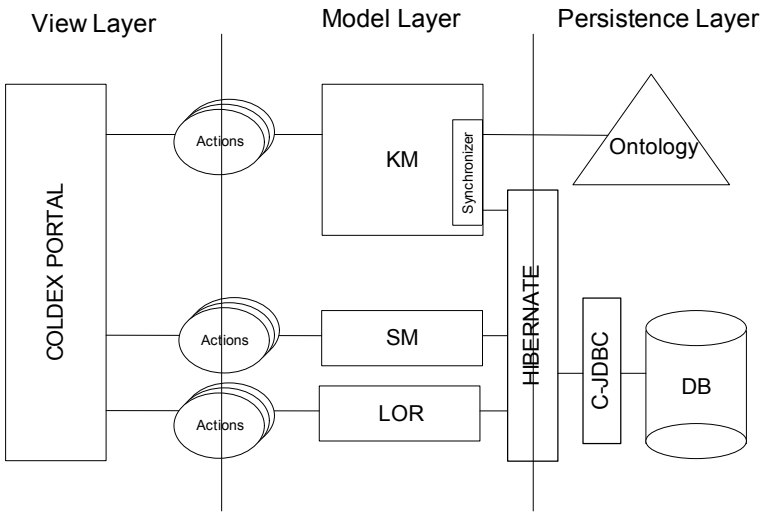

Figure 1. Architecture of the Coldex System

An example of the way in which the Coldex System will be used can be seen in the Open User Scheme, where users from South America will collaborate to work together, in a distributed fashion, to undertake experimental activities together.

As can be seen, the architecture is made up of 4 components, the Coldex Portal, the Knowledge Manager, the Social Manager and the Learning Object Repository (LOR), which will be detailed next.

The Coldex Portal

The portal is a web application that enables users to interact with all modules in the Coldex architecture. As such, it offers individual workspaces to users and operates as an interface to working groups and learning objects stored in the LOR. It has been implemented using the struts framework (http://struts.apache.org/) and uses a set of templates developed using Velocity (http://jakarta.apache.org/velocity/) that enables Web pages to be generated dynamically.

It provides access to the LO repository and offers services to its users. It can be seen to be a communication and coordination medium for the virtual communities formed around it, achieved by the products that handle the LOs, using the workspaces as the underlying infrastructure. The workspaces function as the interface between the users in the virtual communities, the scenarios, and the LO in the repository. The aforementioned services refer both to the creation and maintenance of the virtual communities and the interchange of LO and projects.

The Knowledge Manager

This module allows users to interact with knowledge stored in the protégé ontology, which can be used to separate the system knowledge from the actual code that implements it. Furthermore, this separation minimises the cost of introducing changes into the knowledge in the system and facilitates the personalisation of concepts held in the system.

The Knowledge Manager Component (KMC, for short) exploits the knowledge within the Coldex Portal. Its main purpose is to take advantage of the ontological model underlying the Portal. The system dynamics-in the sense of services, user interface and customisation-are managed by the Social Manager Component. Hence, its behaviour is adapted to the actual groups using the system.

An ontology is an explicit specification of a shared conceptualisation, the use of which for describing declaratively a conceptual model provides the KMC with a greater flexibility as a) the code is separated from the model, which, in turn, b) minimises the cost of making changes, c) the functionalities are not “frozen” in the current code and, so, d) fosters the customisation of the concepts within the model.

The KMC is designed to manage this ontological model. This means making use of the complex knowledge retrieval capabilities that the ontology offers to cope with the

specific necessities of the Portal users. It encloses the communication with the ontology but takes it apart from the user (the calling program, like an Struts action) isolating the internals of writing complex queries from the software using the KMC. Furthermore, it communicates with a query repository, which collects a variety of useful retrieval possibilities.

The KMC offers a public interface, *KnowledgeManagerService*, which defines the *search* method. It incorporates a generic search engine, which specialises into two particular engines, which are able to deal with the queries posed in two available languages. There is also a generic query class with subclasses corresponding to the syntax of the referred languages. The class diagram also shows that the KMC uses the Query Repository (which stores generic queries as a response to the KMC needs) and the Learning Object Repository.

As we have written before, the KMC uses an ontology for describing the system knowledge. The main idea is having the full potential of such a representation artefact in order to enhance the retrieval possibilities already available for the LOR. We have chosen Protégé as our knowledge editor[1] because of its versatility and ease of use.

The Social Manager

This module stores all the information related to social aspects such as the users that can connect to the system, the organisation of groups, previously active sessions. Furthermore, it enforces the policies related to user roles within the system and records accounting data about user access, tool usage, etc. The objectives for the Social Manager Component are the following:

- Administrate the system users, register new users and delete old ones.
- Manage the working sessions of each user (login / logout)
- Manage the social model specified by the system administrators in a declarative form (XML). This will include information such as social structures that can be recognized within each model and the actors that make up one or more social structures.
- Manage the creation and deletion of the social collections in agreement with the socially specified model.
- Manage the relation between the real users and the defined actors in each social situation.

This component is configured in terms of the XML that defines the structures that it manages, i.e., users, sessions and instances of social structures (communities, collectives, groups). The XML specifies a model of a society, and in this example, a collection of communities. Each community is composed of a collection of groups. The society has an actor: “Manager”; each society has an actor: “teacher”; and each group has several actors: “student”. An example of such a society can be seen in figure 2.

[1] We are using Protégé 2.0.1, build 168. Furthermore, we are working with two plugins for reasoning, Protégé Axiom Language (PAL) and Algernon. All the relevant information can be found at the URL http://protege.stanford.edu

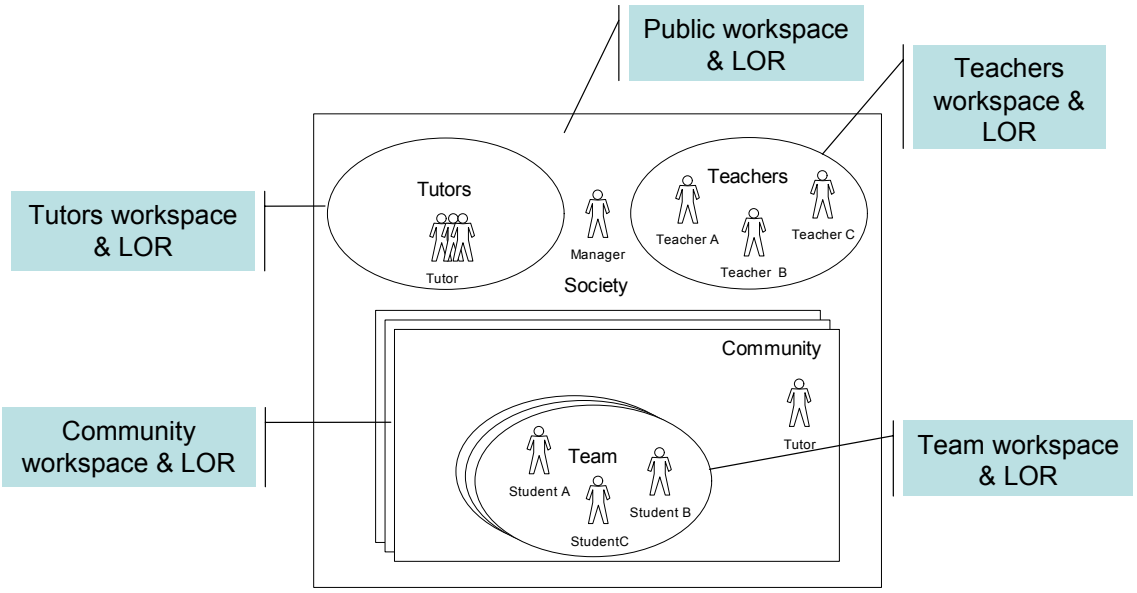

Figure 2. An example society

The Learning Object Repository

This module stores the LOs and manages their access and physical location within the Coldex server, and consists of two parts, one that provides the Java interface that enables objects to be stored, retrieved and deleted, and another that manages the types of objects and associated metadata schemes. The LOR communicates with a database to handle the low level storage. The functionality of this component can be classified as follows:

- Add objects of a determined type. Files can be uploaded.
- Metadata can be provided for the objects with assistance from the system. If an object follows a standard encapsulation format, then the system can automatically extract the metadata.
- Delete objects from the LOR.
- Search for objects based upon certain metadata. Given that the system stores each object in the repository together with its metadocumentación, the user can undertake searches based upon certain metadata. To do this the system provides a Web formula where the user can partially fill in to limit the results that are provided. Searches take into account the context in which they are undertaken (for example, it is not the same to search the repository within one social context than another.), where the semantic relation between objects is analyzed so that objects are returned that have a high degree of similarity (based upon their metadata). The estimation of similarity is based upon production rules that infer the degree of proximity between learning objects based upon certain parts of the ontology.
- Download objects from the LOR.
- View the contents of the LOR without having to download a learning object. The way in which the object is presented depends upon its MIME type.

Persistence Layer: Hibernate and MySQL

The database MySQL is the underlying storage mechanism for the Coldex system, where a relational map stores all the information related to the contents of the LOR. The access to MySQL is handled in a transparent way using a technology called Hibernate (http://www.hibernate.org/), which is a tool that provides a Java API to map objects onto relational structures in an automatic way.

There are two different levels of distribution in which Coldex System can be configured: Partial System Distribution with Centralized Database and Complete System Distribution with C-JDBC.

Partial System Distribution with Centralized Database

The first level of distribution can be achieved using a centralized database. All Coldex servers may connect to the same database, so there may be different web servers sharing the same centralized data. If one of the Coldex servers goes out of service, it would still be possible to keep using Coldex System just by accessing a different Coldex server.

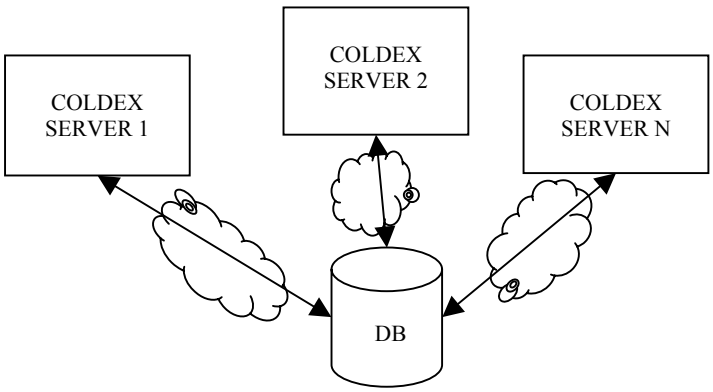

Figure 3. Partial System Distribution with Centralized Database

Complete System Distribution with C-JDBC

C-JDBC (http://c-jdbc.objectweb.org) is a database cluster middleware that allows any Java™ application (standalone application, servlet or EJB™ container, ...) to transparently access a cluster of replicated databases through JDBC. C-JDBC controllers use a communication middleware to synchronize updates in a distributed way.
Hibernate communicates with MySQL through a standard JDBC connector. Because of this, distribution in the Coldex system is automatically granted just by connecting Hibernate software with the MySQL database using C-JDBC:

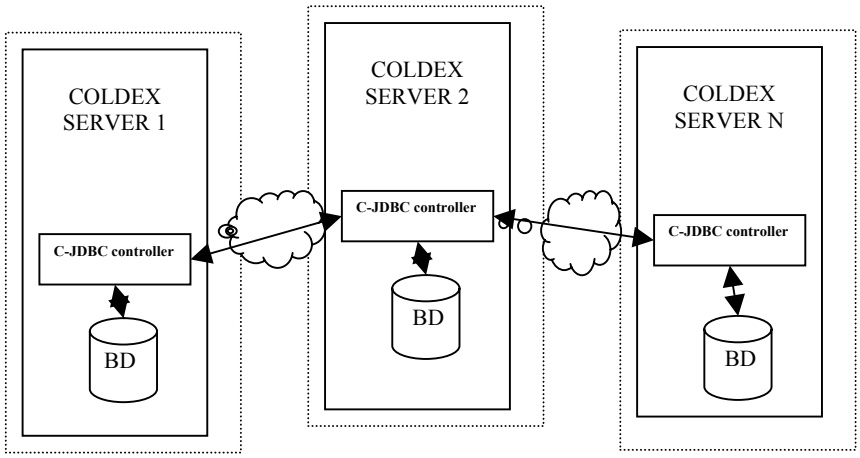

Figure 4. Complete System Distribution with C-JDBC

Installation

The installation process is made up of the following six steps:

1. Install MySQL v.4.0.23. (This is the version that has been tested with the system but other newer versions should present no problems)
2. Define the table structure for the database using the provided SQL script.
3. Install Tomcat 5.0. (This is the version that has been tested with the system but other newer versions should present no problems)
4. Copy the Coldex Web application, coldex.war into the tomcat /webapps/ directory and start tomcat. The coldex.war file includes all necessary libraries for the system to work with Hibernate, Protegé, Algernon… and all the other middleware Coldex system uses, so none of these software is to be installed in a separated way.
5. Edit the /webapps/coldex/WEB-INF/classes/hibernate.properties file to specify the connection details to MySQL database, changing variables:
   - hibernate.connection.url -> Url to database
   - hibernate.connection.username -> Database user name
   - hibernate.connection.password -> Database user password
6. Start a Web browser and connect to the address http://<host>:8080/coldex, where <host> refers to the DNS name of the machine where the installation was made.