

# ESA CCSDS PANEL 2 CONTROL AUTHORITY OFFICE SYSTEM (CAOS) USING XML TECHNOLOGY

Nestor Peccia <sup>1</sup>, Carlos Guerreiro <sup>2</sup>, Martine Julien <sup>2</sup>

<sup>1</sup> ESA/ESOC, Robert-Bosch-Strasse 5, D-64293 Darmstadt, Germany  
Tel. +49 (6151) 902431, Fax +49 (6151) 903010, e-mail: Nestor.Peccia@esa.int

<sup>2</sup> CS Systemes d'Information, Rue Brindejonc des Moulinais BP 5872, 31506 Toulouse, France

## ABSTRACT

Due to the large quantities of space science data being produced by today's space data systems, it is often the problem that the format and meaning of the data is not sufficiently well documented to allow users, who are not directly involved with the systems, to access and use the data. This is particularly the case where the data archive is long term, and people with detailed knowledge of the data are no longer available. In order to address this problem, the Panel 2 of the CCSDS has developed the specifications of the Control Authority Office (CAO) for ensure that data descriptions will always be available (giving a "long term access") to users in an easy manner.

The CCSDS has produced a technical Recommendation for the standardisation of the Control Authority organisation for supporting the digital transfer of space-related information in an open system data interchange using Standard Formatted Data Units (SFDUs). The objective of the Control Authority organisation is to provide a data description management infrastructure to register and disseminate data description (syntactic and semantic description and also data dictionaries) information.

ESA / ESOC took the leadership to develop a CAO system, based on the CCSDS specifications, that would provide automated means for remote users to register, revise and disseminate syntax and semantic descriptions of satellite data via the Internet. It supports two different types of interfaces: a World-Wide-Web (WWW) based prototype of the Control Authority Office (CAO) system that complies with the CCSDS specifications.

a machine interface to the CAO System. In this scenario, also client application software (and not only a human) can access CAO system through an Internet TCP/IP socket interface and retrieve data descriptions on-line for automated processing.

The system currently running at ESOC is based on old technology and supporting several missions (e.g. ERS 1-2, Huygens, XMM, Rosetta, Mars Express, Smart 1).

The advent of the Extensible Mark-up Language (XML) has triggered ESOC to analyse the possibility to replace the current CAOS by other one based in XML technology but maintaining the same functionality and adherence to the CCSDS P2 standards. This work is being carried out within a more global analysis (from CCSDS) about the use and the advantages of XML in space related applications.

XML moves the control of the information out of the hands of others and into our own by providing two basic facilities.

XML describes rules for structuring information using embedded mark-up of our own choice.

In addition, XML describes a language for formally declaring the vocabularies we use.

The paper will describe the XML based CAOS architecture by

- identifying the functions that could be implemented using XML technology.
- enhancing the current CAOS system by linking it with the data descriptions produced by a Data Description Language (e.g. CCSDS EAST)

- Assessing how the data defined in the current CAOS can be exported and then imported into the new XML based CAOS

The paper will also report about the approach adopted on the prototype and the lessons learned for the final implementation. This paper presents the summary of the CAOS XML prototype development and the architecture of the future system based on the lessons learned through this prototype.

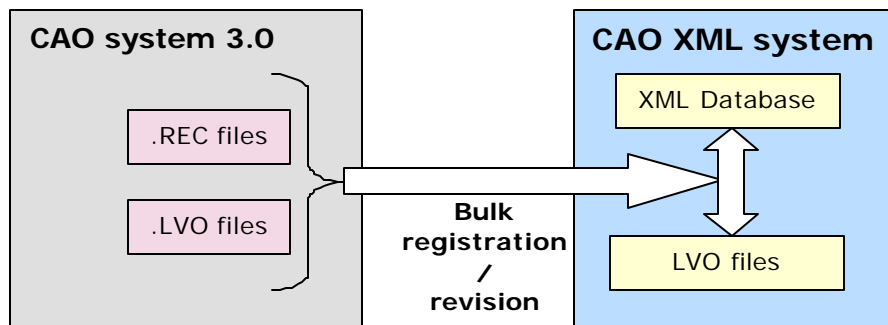
### Introduction

The primary goal of the study was to determine the ways to enhance the current system using XML technologies. It has been obvious that the system was quite old concerning the tools used (e.g. versions of the operating system and of Perl, Perl/XML world is very poor in comparison with the Java/XML community). It has been chosen to abandon Perl development and use the Java programming language instead. Java is favoured in comparison with Perl since it is in a mature state and also insures code portability. Java also has a natural ability to deal with XML-based information. This prototype is then a complete re-elaboration of the existing system.

This prototype provides three main functionalities:

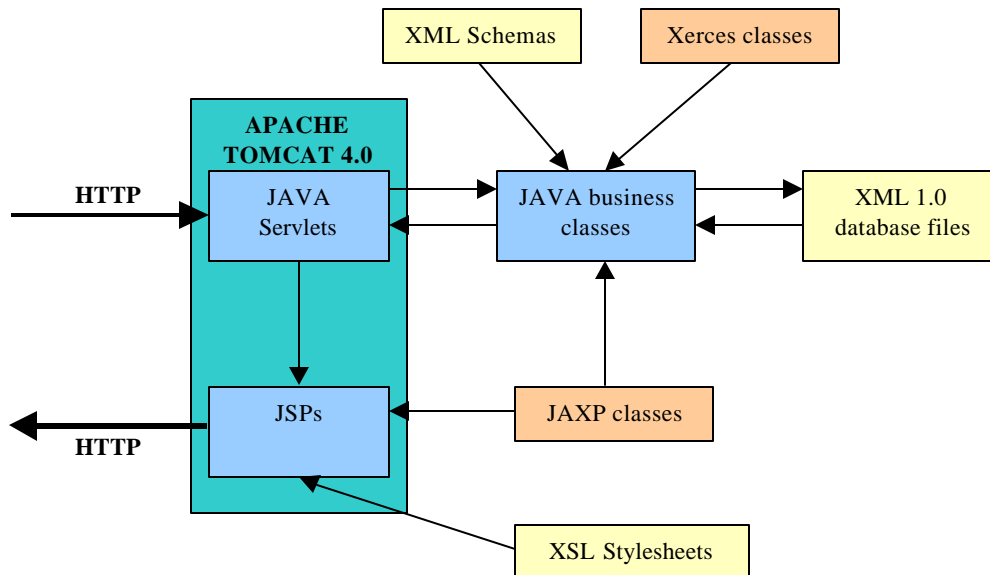
- data descriptions registration,
- data description revision,
- and data description dissemination.

Furthermore, to ensure data descriptions perennially, an administrative tool (see figure below) has been developed in order to populate the prototype with old database files (.lvo and rec data files) extracted from the existing system (version 3.0). The lvo and rec files are created as a result of the complete dump functionality of the CAOS 3.0 system. In this prototype, XML has been used for all configuration and database documents. Java is then used to handle client requests and responses (through the use of servlets and JSPs) and perform the associated actions (register user/data description, ETC.).



**Figure 1 CAOS bulk registration process**

The figure below details the use of Java and XML in the prototype:



**Figure 2 CAOS XML prototype architecture**

All blue boxes represent the Java development. The servlets and JSPs are managed and executed using the Apache Tomcat server. The yellow boxes represent all XML files. Schemas are used to validate both users and data descriptions database files. XSL stylesheets insure users database information transformation into HTML tags to be included in JSPs. Orange boxes represent all base classes. JAXP is delivered by SUN in the latest release of the JDK (i.e. version 1.4.0) while Xerces is an Apache project.

As shown in the above figure, the prototype is used in the following way:

1. A client sends a request to CAOS.
2. This request is then caught by CAOS servlets and transformed into actions. These servlets will control all actions executions.
3. These actions are performed by the Java business classes.
4. When finished, the servlets store results and pass them on to JSPs for display.
5. A response is then produced and sent to the client.

The first comparison of Perl/XML world and Java/XML community has shown that XML has to be used with new technologies to be simply and correctly used. It has also highlights the fact that CAOS (in its third version) was running on too old tools releases using technologies that are nowadays obsolete.

Then, the test of the version 3.0 of CAOS has exposed the difficulty to upload data descriptions components. This functionality was previously processed using the FTP (File Transfer Protocol) protocol. However, there is an easier way to upload files by using the HTTP (Hypertext Transfer Protocol) protocol.

The following problems have been encountered while manipulating XML documents:

- Developers may take a particular attention to the tool they use to edit the XML documents. XML documents tend to be very large. Then, navigation through the XML elements will be very tedious.
- JAXP (Java API for XML Processing) was used as interface between Java classes and XML documents. The main problem has been the validation against XML Schema. The latest release of

JAXP only supports validation against a DTD. In order to use the schema validation we had to use the Xerces classes.

- Furthermore, parsing large files may generate performance problems.

On the contrary CAOS has benefits from using XML and Java:

- Leveraged portability. On the contrary, endianness doesn't matter in the case of XML database files.
- Leveraged maintainability by using an object oriented programming language.
- Increased performances.
- Leveraged CAOS openness since new technologies like web services for example can be easily integrated in CAOS. Furthermore, CAOS could be used as a plug-in in the DEBAT tools suite for data description management. DEBAT is a complete re-elaboration of tools based on the EAST technology which offers means to design and use EAST data descriptions and associated data products. On the other side, CAOS users could benefit of DEBAT functionalities through the use of the DEBAT web services to:
  - Read data file content using an EAST data description,
  - Generate data files given an EAST description file,
  - Extract relevant information from data files.
- Leveraged compatibility between CAO XML systems.
- Leveraged independence from any operating system and programming language. In fact, using XML as database format, the information can be accessed and manipulated using any programming language. It is an additional guarantee to insure perennially the data descriptions.

### Definition of future CAOS System

The existing CAO system (version 3.0) is assumed to run on a POSIX compatible environment. Thus, the XML CAO system shall run at least on a POSIX compatible machine. However, as the XML system has been developed using the Java programming language, CAOS should run on any platform (provided that a Java virtual machine does exist) with no or few changes.

Like the existing CAO system (release 3.0), it has been assumed that dissemination is the most important, followed by registration and revision. With regard to the interfaces, the WWW interface is considered more important than the software and the web services interface. The system shall be developed using open source and free software. The future system will provide the same functionalities as before:

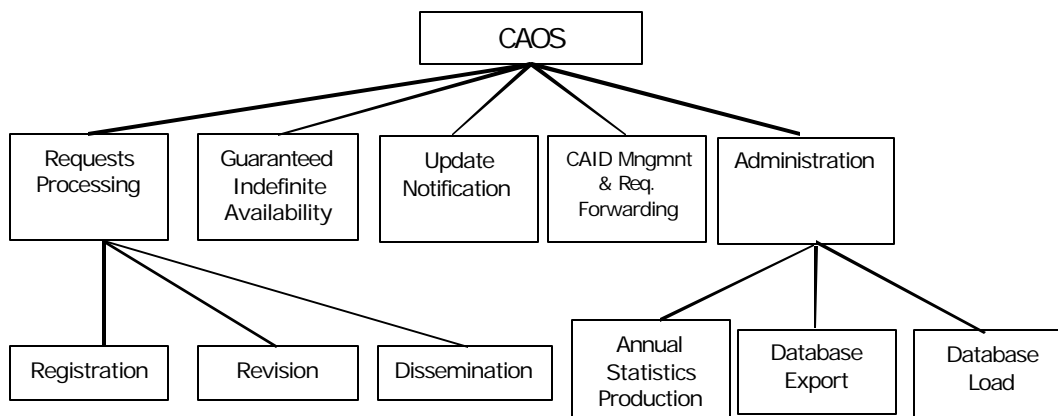
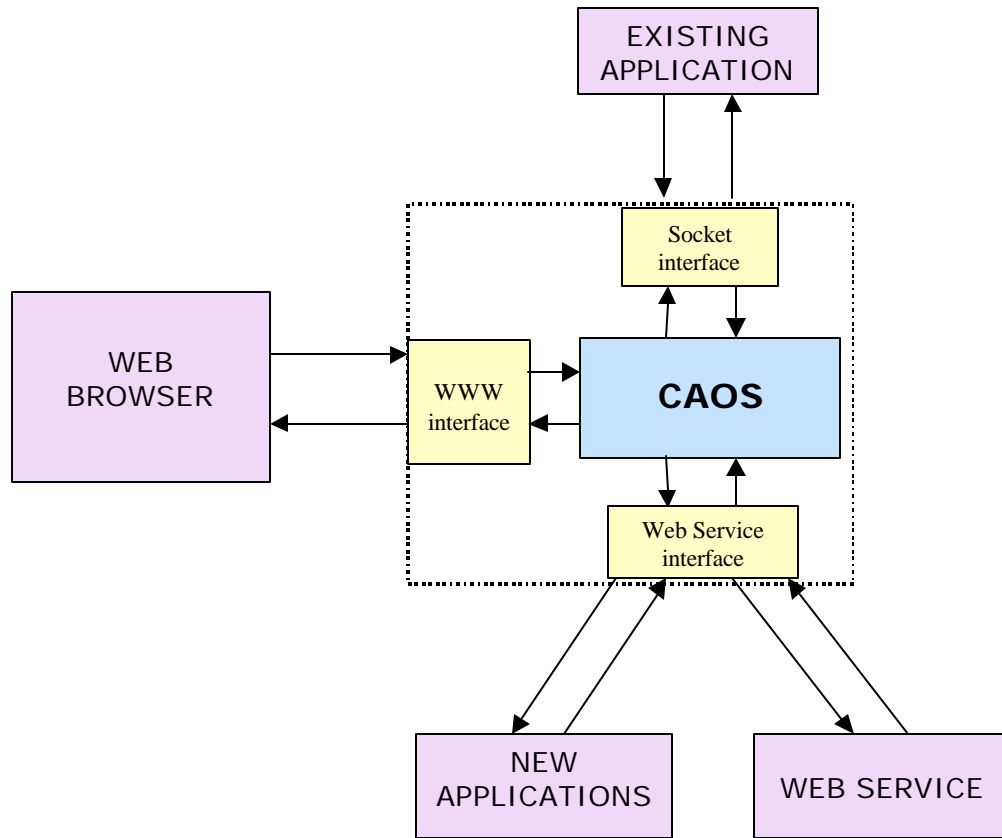


Figure 3. CAOS key services

### Interface Specifications

The system will offer three distinct interfaces:

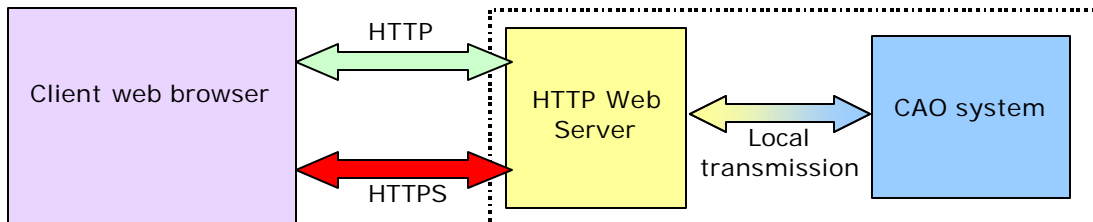
- A World Wide Web interface.
- A Web Service interface. This interface comes as replacement of the socket interface for all new applications and others web services.
- A socket interface.



**Figure 4. CAOS Interfaces**

**The WWW Interface**

This interface shall be used by any users having interest in data descriptions and also by the administrator.



**Figure 5. CAOS WWW interface**

**The socket Interface**

This interface resumes the functionalities of the existing socket interface.

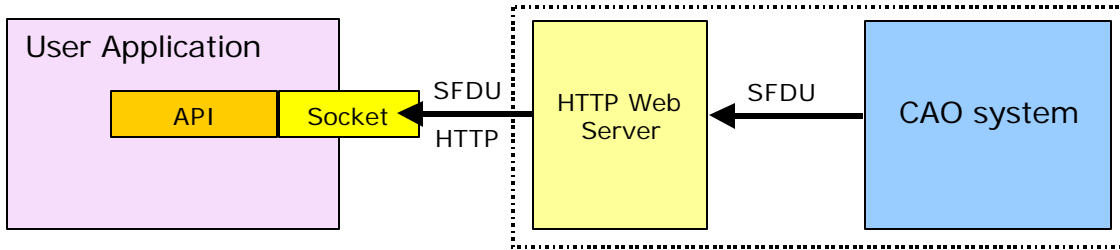


Figure 6. CAOS socket interface

The data descriptions are delivered **over the HTTP protocol** using the **SFDU standards** for encapsulating the data descriptions.

### The Web Service Interface

The web service interface will also only support the dissemination functionality.

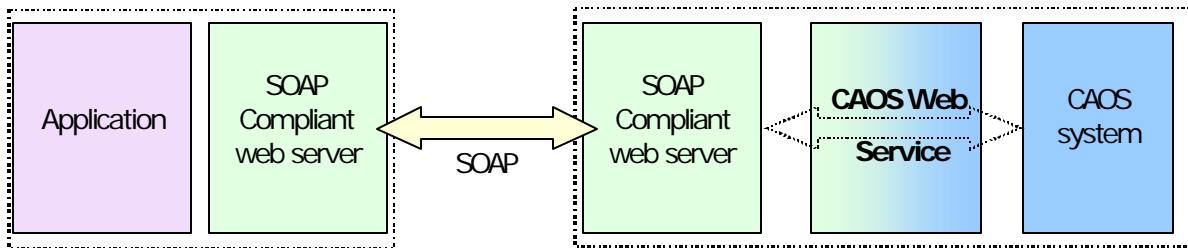


Figure 7. CAOS web services interface

The data descriptions are delivered **over the SOAP protocol** using **XML** for encapsulating the data descriptions information.

As the web service interface is generic and offers access to CAOS functionalities, the CAO system can be easily integrated into any other applications to be used as an archive tool. For example, it could be used as a part of the DEBAT project.

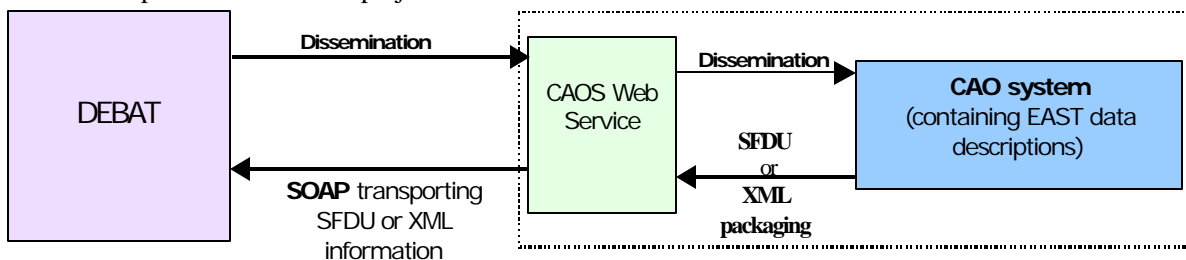
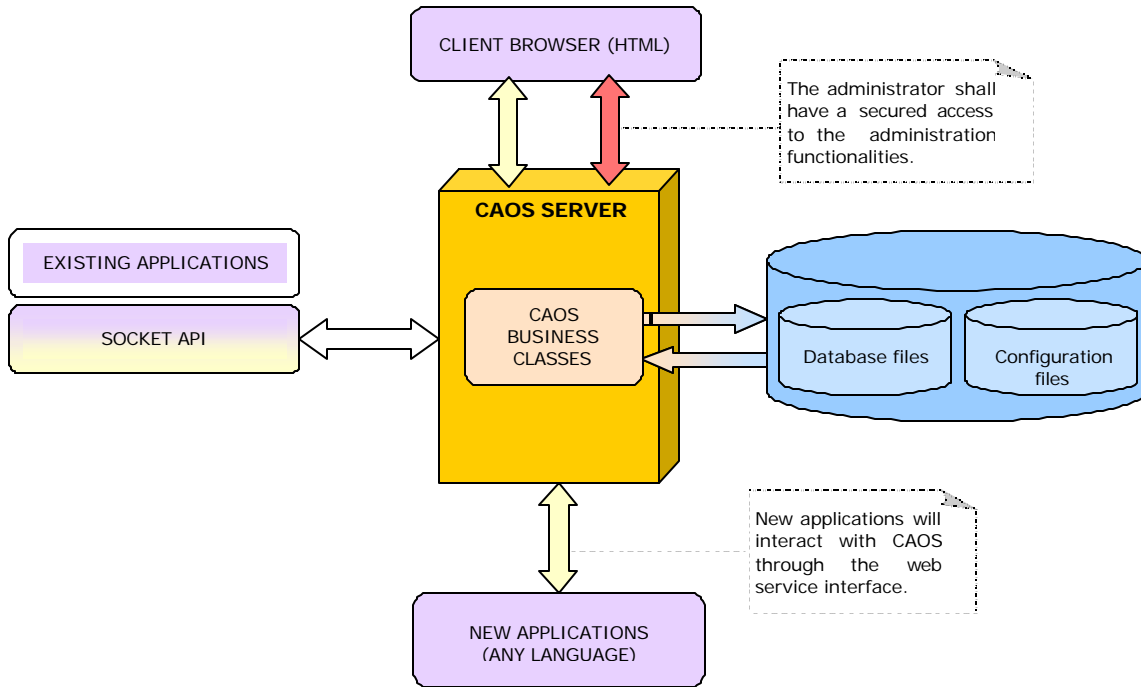


Figure 8. CAOS use example within DEBAT

### Architecture

The following figure gives an overview of CAOS architecture and explains how the system will interact with its users.

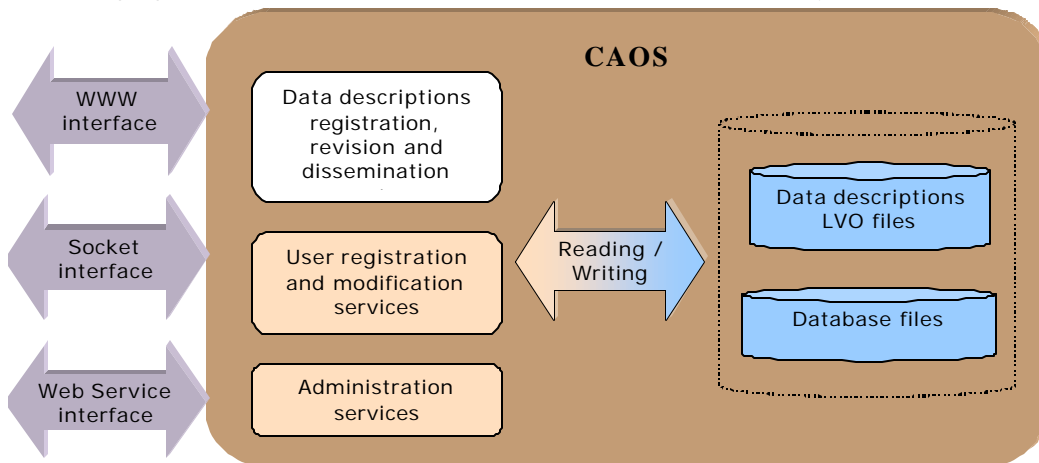


**Figure 9. CAOS Architecture overview**

This figure shows a well-known 3-Tiers architecture that is easily maintainable and updateable. Each tier can be easily modified without impacting the two others.

### Functional Architecture

The following figure details the functional architecture of the future CAOS system:



**Figure 10. CAOS functional architecture**

There are three main functional areas:

- Data descriptions management (registration, revision and dissemination). All of these functionalities are accessible through the WWW interface whereas only dissemination is available for socket and web service interfaces.

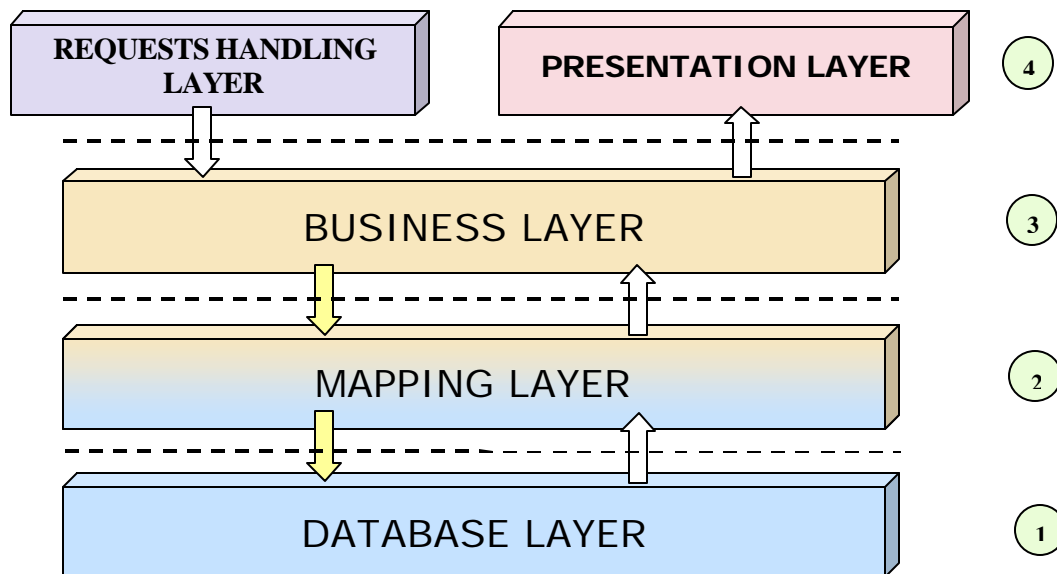
- Users management (registration and profile modification). These functionalities are only accessible through the WWW interface.
- Administration services (annual report creation, bulk registration/revision,...). These functionalities are accessible through the WWW interface using a secured access and only for the administrator. This latter can also administrate the system thanks to command lines directly on the server machine.

### Software Architecture

To be easily developed and maintained, CAOS has to be divided into four layers:

1. One bottom layer corresponding to the database called **Database Layer**. This database will contain all information concerning data descriptions and users. The data descriptions components will still be stored as LVO files outside the database.
2. Another layer called **Mapping Layer** to read and write information contained in the database files. It will insure the mapping between data stored and CAOS objects. Furthermore, developers won't have to know the database structure.
3. Another layer called **Business Layer** to execute business process based upon clients' and database information.
4. One top layer to act as an interface with user divided in two parts:
  - the **Request Handling Layer, which** will handle all clients' requests and transform them into actions to perform. It will also insure that all these actions will be performed.
  - the **Presentation Layer**, which will render all results to clients.

The system will then be divided in four separate layers:



**Figure 11. CAOS layer architecture**

The two top layers will be developed using the Java programming language:

- The requests handling layer will be implemented using servlets and web services.
- The presentation layer will be represented by JSPs (used in the case of a WWW interface).
- The business layer will be developed as Java objects.

There are four possible solutions for the database and mapping layers:

1. Save information as XML documents as done in the prototype. JAXB (Java API for XML Binding) could then be used to automatically create the mapping between Java classes and XML documents. However, this method has many drawbacks (see chapter **Error! Reference source not found.**).
2. Save XML elements in a native XML database. Such database enables collections of XML documents. Here information concerning a sole data description will be gathered in a document. Then, using collections, these descriptions could be gathered by MACAO. Furthermore, all XML documents in the database have an associated key. The ADID of the data description could then be used as key to facilitate its retrieval.
3. Save all information in a relational database and access the data using JDBC (Java DataBase Connectivity) to create the relational-object mapping.
4. Save all information in a relational database and create XML views of the stored data. Queries could then be processed upon these views using an XML query language

### Conclusions

The prototype has demonstrated the feasibility of XML technologies in the system. However, this system only implements a subset of the desired functionalities and should be enhanced to take the whole users requirements into account. Furthermore, XML technologies are evolving rapidly. Many tools and norms based on XML are appearing that could leverage the CAO system (e.g. the XQuery language).

The CAO system has reached its limits because it hasn't been designed in order to accept the changes required by the apparition of XML technologies.

Moreover, the XML prototype has demonstrated the feasibility of XML for data descriptions management. One important thing here is that XML recommendation (i.e. version 1.0) is fixed whereas XML tools always evolve to be more powerful. By taking advantages of these tools, CAOS could then become faster and faster while keeping the same XML structures to store information.

Furthermore, the apparition of XML querying languages like XQuery will facilitate the creation of statistics and even create new sets of information while querying the whole database to extract some precise information.

Another point is that XML is open and leads to the creation of many standards. The main problem here will be to distinguish which of these standards are useful for CAOS and which of these will receive the support of the industry. In this area, Apache projects (like XIndice) have always been widely adopted and, though free, have always been reliable.

In CAOS, XML can then be used:

- to enhance the client side (through SVG and XSL for example),
- for communication with other applications using web services,
- to insure data descriptions perennially using XML structures to store information,
- to store configuration information.

Relational database are nowadays completely reliable and secure. However, they may not be capable of evolving to offer much more functionalities to users and developers.

On the other side, the use of XML fixes definitely the format of the data. It means that once data are stored in an XML format, they can be accessed and manipulated using any parsers or tools (existing or to come) and this independently of the platforms and operating systems. Moreover, the XML world keeps growing to offer increasingly more advanced functionalities (e.g. XQuery language, XML native database). XML is then a perennial format since data structures don't evolve through time. Only tools and standards used to manipulate these structures change to get more and more powerful.