Java-based User Interfaces for CERN’s Engineering Data Management System.

Bertrand Rousseau[1], Christian Luong[2], Bruno Faugeras[2], Christophe Delamare[2], Thomas Pettersson[2].

[1] CERN, IT Division.
[2.] CERN, EST Division.

Abstract

In the framework of the CEDAR project, CERN is implementing a commercial Engineering Data Management System (EDMS): CADIM from Eigner & Partner. CERN’s EDMS will be used to store, organise and control all engineering data for the LHC and its experiments : drawings, text, etc., throughout the design, construction and exploitation of the LHC; more than 20 years.

Customisation of an EDMS is a long and heavy process, that covers requirements analysis, data modeling, scripts writing, integration with external software, etc. The Toucan project takes part in this customisation effort by developing user interfaces for the EDMS, in close collaboration with the EDMS vendor. These interfaces include:

- A high level Java Application Programming Interface based on an Entity-Relationship Model. This interface is EDMS independent and allows not only to communicate with CADIM, but also with other CERN databases.
- A Toolbox of Java / Swing graphical components, built on top of the Entity-Relationship API. These components can be configured and assembled to build graphical EDMS clients tailored to certain kinds of tasks and users.
- An Import / Export format based on XML that allows massive batch exchange of data and metadata between the EDMS and external software.

After a brief presentation of the CEDAR project, we focus on the requirements, design and implementation of the Toucan user interfaces.

1 CEDAR - An Engineering Data Management System for CERN

The design, construction and operation of the LHC and its experiments will generate a huge, complex and multi-disciplinary collection of data. This data cover documents and pieces of information such as CAD drawings, technical reports, product configurations, catalogues of components, simulation and analysis results, calibration data, software source code, meeting minutes and slides, detectors images, project plans, etc. Data production will be performed in many different institutes, located all over the world using a whole variety of methods and tools, ranging from word-processors to CAD systems. The need for ease of access, for consistency and maintenance, for 20 years availability and for rational organisation of the work with such a large and complex amount of data dictate the use of an Engineering Data Management System, an EDMS [1].

An EDMS has numerous functions to manage data and documents, process and workflows, product structures and configurations, classifications, etc. It is built around a database (often called a data vault), in which all kinds of data used to define, manufacture and support products are stored, managed and controlled. An EDMS stores not only document files, but also the so called metadata, objects that give structure and semantics to the set of controlled documents. For instance, the various components of a detector can be modelled as a tree shaped *Product Breakdown Structure* (PBS) ; Documents objects record information such as authors, status, format, etc. (See Figure 1). More details about EDMS can be found in [2].

Started in 1995, the CEDAR project aims at implementing an EDMS at CERN. After a long phase of market survey, products evaluation and pilot projects, a commercial EDMS was finally purchased: CADIM / EDB from Eigner + Partner [3]. The role of the CEDAR project is now to customise CADIM in order to adapt it to CERN’s needs. Customisation is a long and difficult process that covers tasks such as the design of a data model for CERN’s metadata, the definition of workflows, integration with software tools (e.g. CAD and desktop) and existing legacy databases and more generally embedding the EDMS into CERN’s work procedures. Current customisation effort is mostly spent in the field of the engineering data for the LHC accelerator and its experiments.

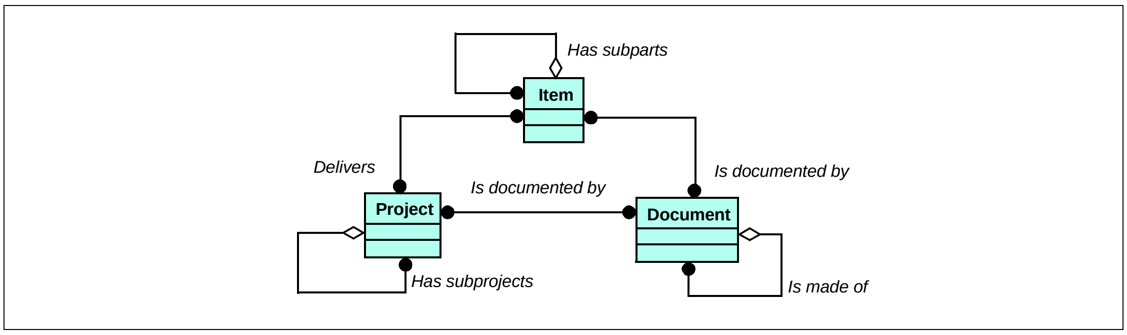

Figure 1: The heart of the EDMS data model is made of 3 interrelated entities. Items represent parts of an equipment to be built. Items can be made of other Items. Items are documented by a number of Documents that corresponds to physical files. Items are delivered by Projects.

2 Requirements for EDMS User Interfaces

An essential part of the customisation process concerns the development of the user interfaces. The native graphical interface of CADIM (and this seems also true for other commercial EDMS) is too complex and not flexible enough to accommodate the variety of users and tasks in CERN's environment. For instance, a majority of users will mainly use the EDMS to obtain documents and / or to store documents they produce. This category of users require database navigation and document check-in / check-out functionality packaged into simple abstractions. Other users such as project managers need more complex interfaces to build and maintain metadata related to equipment assembly structures.

Today, most users access the EDMS through a unique interface: the CEDAR Web Interface [4]. The approach of the *Toucan Project* is different. *We aim at developing a software infrastructure and abstractions to build, deploy and maintain dedicated EDMS clients for user interfaces and applications that need EDMS access.* A number of important requirements have guided the design of the Toucan software: scalability, portability, flexibility, coverage of EDMS functionalities, etc. Some of them resulted in the choice of Java as the Toucan programming language. Three of these requirements are detailed below, because their fundamental impact on the software design.

1. **Adaptability to an evolving data model**. The EDMS clients should not depend on changes in the data model. The data model is made of entities, relations and relational tables that will evolve along the life cycle of CERN's EDMS. If the major objects that constitute the heart of the data model (see Figure 1) are well known and are not likely to disappear, new fields may be added to them, and new entities may be created. These changes must not affect EDMS clients.
2. **Ability to support future migrations**. More drastic changes can be also be foreseen. The dismantling of the LHC is so far away and the domain of information systems is moving so fast that we can bet on future migrations to other EDM Systems. As a consequence, the user interfaces shall be designed to be independent of the underlying EDMS.
3. **Ability to cope with poor network connections**. The EDMS clients shall be efficiently usable even for far-distance users with slow network connection.

3 The Entity Relationship Layer - A high Level Programming Interface

CADIM / EDB is delivered with an Application Programming Interface called ECI, that allows socket-based communication between a client and CADIM. Not only this interface is EDMS dependent, but it is also data model dependent since different functions must be called in order to perform similar operations on different kinds of objects.

Our first step was to develop a high level programming interface that hides this low level complexity by providing a simple and regular Entity-Relationship model supported by high level Java classes and methods. This Entity-Relationship layer is the foundation on top of which client EDMS applications can be built. Such applications are numerous: graphical user interfaces, batch programs to perform maintenance operations for examples.An example of Java source code using the ER Layer abstractions is shown in Figure 2.

```
// create a new Item record
EntityRecord item = LocalDataManager.newEntityRecord ("Item");
// assign value to some of its fields
item.setValue ("Name", "Octupole Corrector Assembly");
item.setValue("Project Engineer", "Arthur Martin");
// insert it in the EDMS
item.insert ();
```

Figure 2: This small Java program excerpt shows some operators of the Entity-Relationship Layer.

A important design issue concerned the independence of the ER Layer with respect to the CADIM data model. Because this data model will be in constant evolution, and in order to avoid a huge maintenance work, the ER Layer shall not integrate any knowledge about the entities and relationships stored in the EDMS: names of entities and fields, names of corresponding CADIM tables and masks, etc. For instance the ER Layer shall not assume the existence of Java classes named Item, Document, etc. Consequently, the data model must be loaded and integrated at run-time in the client application. The ER Layer must thus provide data structures to record this data model.

The solution we have retained consists in writing EDMS dependent Java classes representing Items, Project, Documents and their inter-relationship. These classes are located on an http server where they are centrally maintained. The client application loads these classes at run-time, and is able to use them through well defined Java interfaces (see Figure 3).

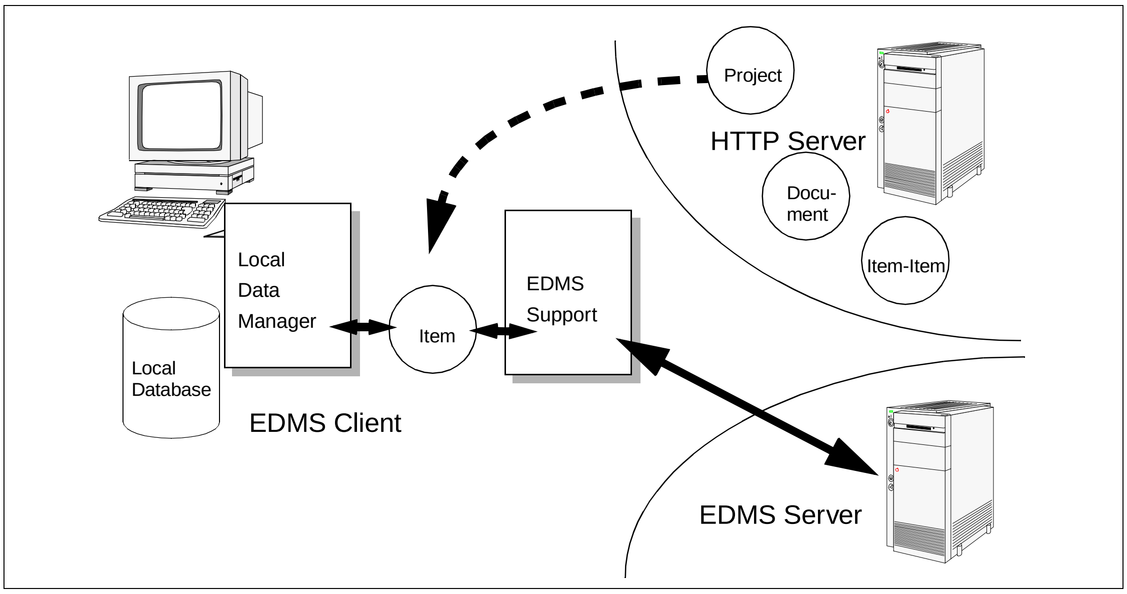

Figure 3: An EDMS client using the Entity-Relationship Layer relies on a Local Database that caches part of EDMS data. This data (modeled as entity and relationship record) is stored as instances of Java classes that are loaded upon request from an http server, at run-time. The Local Data Manager can use these classes through well defined interfaces and through Java reflexivity. Such classes can perform all kind of operations, e.g. accessing the EDMS or other databases, as long as it implements the required interfaces. These approach allows an EDMS client to be independent from the underlying data model and databases.

Efficiency of work, despite heavy communications between the client application and the CADIM server, is one of the main requirements for the design of the Java API. One way of dealing with this constraint is to let client applications access a local database rather than CADIM records directly. At a given time, the local database contains a copy of CADIM records that are of interest for the user. The local data manager communicates with the CADIM server to load new records from CADIM, or update CADIM with local records. The network bottleneck is then limited to the transfer of data between the client and the server. Once data is present in the local database, many operations are performed locally. It is possible to select CADIM records according to a number of criteria, and to load them into the local database. These records may thus be changed and the modification propagated back to CADIM. It is also possible to create new objects (e.g. Entity records or Relation records) in the local database and to insert them into CADIM.

4 The Graphical Interface - A Toolbox of configurable Components

The Toucan graphical interface is build on top of the ER Layer, and is thus guaranteed to stay independent from the underlying EDMS and data model. It consists in a set of graphical Java / Swing components that can be individually configured according to user needs.

Each component provides a specific "graphical point of view" on a set of records. For instance the Tree Component allows to view entities and relation records as tree nodes and branches that can be shrinked or expanded as required. The Table Component presents entity records and the values of their fields as a table. The Form Component shows all fields and properties of an entity record.

These Components appear as windows into a larger EDMS Desktop window.

A important abstraction brought by Toucan is the concept of Bookmark. Bookmarks are named folders that help users to keep and store records of interest during an EDMS session or across multiple sessions. A bookmark can be "opened" in a graphical component, causing its records to be displayed.

Graphical components may subscribe to other graphical components in order to be informed of changes. For instance a Table component can be configured to add new rows when records are added in a Tree component.

5 An Import-Export facility based on XML

Embedding an EDMS in a complex organisation like CERN implies being able to exchange data to and from external tools and databases. For instance, EDMS metadata such as product structures are often edited in the form of MS Excel tables and must then be imported into CADIM, data must be extracted from CADIM to be inserted into other databases, etc. In addition CERN's engineers are currently faced with the problem of populating the EDMS with a huge volume of existing data.

This kind of massive import export cannot be performed "manually" with an interactive tool. This is why we have developed an import / export format that can be generated or read by the Entity-Relationship layer. This neutral format is based on XML and is thus simple enough to be edited manually, and to be generated or parsed by dedicated converters (e.g. Excel to XML).

6 Conclusion

The Toucan project has delivered the second version of the Entity-Relationship Layer. We are now focusing on the development of the graphical interface. The XML import / export format is already in use in several LHC and ATLAS projects and appears to be very useful to feed in the EDMS with existing data and metadata, especially for bulk loads.

Java has proved to be an invaluable tool to implement the Toucan Software. Its portability, its libraries (Swing, XML, net, ...) and its unique features such as dynamic class loading and reflexivity allowed a direct coding of most parts of our design.

References

1 B. Rousseau, N. Høimyr, C. Hauviller, Implementing an Engineering Data Management System for the LHC Accelerator and Experiments: The CEDAR Project, *Computing in High Energy Physics*, Berlin, April 7-11, 1997.

2 CEDAR Home Page: *http://www.cern.ch/CEDAR/*

3 Eigner + Partner Home Page: *http://www.ep-ag.com/products/*

4 CEDAR Web Home Page: *http://edmsoraweb.cern.ch:8001/cedar/search*