Building Controls Software around an Object Oriented Database

Kris Kostro
CERN, SL Division, CH-1112 Geneva 23, Switzerland

Abstract

The use of Object Oriented (OO) techniques has become popular in all areas of software technology and HEP control systems have not been excluded from this trend. In the course of modernisation of the CERN SPS Experimental Areas control software we designed and implemented an OO database to hold the configuration data for equipment and beams. With the beam lines and equipment defined in the new database, control facilities are being added by incrementally enhancing the classes and adding new methods to the database schema. Using the OO database helps to design the new system in a transparent way. Real-world objects such as beam lines or crates are uniquely mapped to the corresponding objects in the database. The new database allows seamless integration of data into programs written in OO languages such as C++ and Java. The WWW interface to the database gives a familiar look and feel and has been provided with relatively little effort.

In this paper we present an overview of the project and the employed methods. The choice of the database management system, the implementation of the Beam Instrumentation (BI) database and its use in controls as well as the benefits of the approach will be discussed in some detail.

1 Experimental areas of SPS and their controls

The secondary beam lines of the SPS at CERN are located downstream of the SPS primary targets: T1 in the West Area and T2, T4 and T6 in the North Area. These beam lines serve a variety of experiments and test facilities often installed for several years or just for a few weeks. The beam lines have therefore to be adapted continuously to the changing needs. To accommodate for this, the control system has to contain facilities for an easy installation of new equipment even during operation.

The control system for experimental areas provides facilities for controlling the beamline equipment by experiments in an individual manner. Experiments have their individual permissions, which restrict equipment access to their beam line, and other beam lines are not visible to them. Another important function of the EA controls is surveying the access doors in the zone and verifying the position of some critical equipment such as the wobbling magnets.

The original control system of the Experimental Areas dates from 1976, and it was based on NODAL programs running on NORD computers with CAMAC for equipment interface. In 1994/95 the control system was upgraded and a new system architecture, similar to the one used in LEP was introduced [1]. This architecture is often referred to as the " standard model". NORD machines were replaced by HP-UX workstations at the control room layer and by PCs running the LynxOS operating system on the front-end layer, leaving the equipment control layer unchanged. The large investment in hardware at the level of CAMAC and below makes the replacement costs prohibitive so that new low-level equipment control is introduced on case-by-case basis. To avoid rewriting the large suite of application programs written in NODAL over the preceding 20 years the NODAL programming environment was recreated for the HP-UX operating system. This approach has helped to replace the obsolete NORD computers, but the many, often little understood NODAL programs were left in place and this constitutes now the biggest problem for the Experimental Areas controls. The software interconnections of this system are represented at the left-hand side of Figure 1. The right-hand side represents the new software connectivity and will be discussed later.

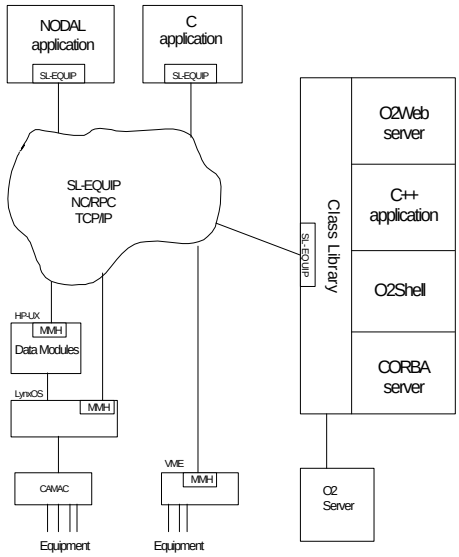

Figure 1

2 Evolution of the EA controls

The large number of programs written in an obsolete language, the little knowledge we have about them and the limited manpower make it impossible to replace all the EA controls in one big project. Since the NODAL programs use their own data structure to store data and interact between themselves heavily, it is also difficult to replace them in an incremental way. Fortunately the project to

replace the Beam Instrumentation (BI) database of connections and addresses has offered us the opportunity to slowly migrate the control system to a new environment. OO techniques employed in this project provide a infrastructure on which the new controls software can be built.

2.1 The Beam Instrumentation database

The database of the electronics modules and their connections has been one of the most complicated pieces of software ever written in NODAL, and it represented an effort of several man-years. This database fulfils several functions in the current system. The obvious one is to keep track of used racks, crates and modules and their interconnections. There are about 5000 physical items in the database. It is also used for creating new equipment according to equipment templates. The system scans the database searching for free channels and allocates them to the new equipment. Calculating addresses of equipment is another capability of this system. Every equipment type has its own addressing program. Modules can be wired together to create a common facility, called *system* and *system* channels can be used in equipment and other *system*s. With about 60 equipment and 30 *system* types (and almost as many addressing variants) the complexity of this database was obvious. With maintenance becoming more and more difficult, the replacement of this database, responsible for the equipment address generation, became an absolute necessity but it has also been taken as a challenge and opportunity to introduce new techniques.

3 New software system for the Experimental Areas

3.1 The new database choice

From the beginning we wanted, if possible, to use an Object Oriented database. The main reason was that, being interested in OO techniques, we wanted a database, which is well integrated into this environment. The ODMG – Object Database Management Group defines language bindings to allow the database to be accessed in a transparent way from OO languages such as C++ or Java. Another reason is that an OO DBMS is inherently better suited to represent complex data structures than a relational DBMS. It is often stated that the performance of an OODBMS is superior for complex data. Recent investigation in the HEP community [2] showed 4-50 times better performance for typical physicist queries. At CERN the Objectivity OODBMS was chosen by the RD45 project for performance reasons and for support for very big databases in the order of terabytes. Our choice criteria were different: we were looking for connectivity, compliance to standards to protect our investment and for development tools.

The system which meets these requirements the best, is O2 from O2 Technology. The offered connectivity is O2Web to access the database from a WWW browser, a CORBA server to access the database from OO applications distributed over the network, and a bridge to relational systems such as Oracle. C++ and Java bindings provide a transparent access to database objects and a C interface exists as well. The database can be queried with the OQL query language from any of these languages or from the O2Shell. OQL allows using class methods in the query (among other features) and is part of the ODMG standard.

The market for OODBMS, although rapidly growing, is still relatively small, compared with the one for RDBMS. Because of this, the price for such systems is still relatively high and there is no well-established market leader. This should change in the near future given that the OO technology will continue to gain ground.

3.2 Implementation of the new database

After installing O2 in November 1996 we started the implementation by defining the classes of the database schema. Some of the classes such as *Rack, Crate, Module* correspond to real-world objects and are obvious to define. Other classes such as equipment template represent an abstract idea and the relation between the BI equipment and it' template is less obvious. Flexibility was the highest priority in designing the classes of the schema and much effort was invested there. We anticipate that the database schema will evolve in the future and it will be used by other applications so that we tried to separate out different views of objects. For instance we separated out the physical, and the control aspects of experimental equipment, which was not necessarily the case before. About 50 classes have been defined of which 20 are key classes, such as *BeamSegment, ExpEquip* or *Crate,* and represent key ideas or real-world entities. Other classes are helper classes or are used merely as data types.

A substantial effort went to the population of the new database and several C ++ programs were written to generate objects based on data extracted from the NODAL database. Another laborious task was understanding and implementing the numerous addressing algorithms.

When implementing a database, the user interface is one of the most time-consuming components. Initially we planned to implement a Motif-based interface for which there is some support from O2 tools. Feeling that Web-based interfaces are gaining on importance, we decided adopt a WWW approach. This is supported by the O2Web tool, which provides the first, rudimentary interface to the database at no development cost. Some customisation is then needed and can be done on a class-by-class basis in an incremental way. With the default interface the objects are identified by their class name which is obviously not very convenient for navigation. A small, one-line method returning the name of the object is normally sufficient to overcome this limitation. Another problem with the default interface is that when navigating from object to object, the URL is extended every time with the object reference. For some databases, where navigating in circles is possible (it is enough to have a back-reference) this can lead to very long URL, hampering the performance. To overcome this a *get_query()* method, with the OQL query leading to this object, can be provided, which will break the circle. On a

typical Web page [1] attributes and references are represented as a list with square bullets. Lists of objects are bulleted with circles. All references are *clickable*.

Another case of HTML generation occurs when customising the default page is not sufficient, i.e. when we want to personalise the look of the object page, point to another Web page or make input to the database with forms. This can be done by customising the *html_header()* method. This way buttons can be easily added to invoke an action on the object or to spawn a form to submit data to the database.

When using forms two methods are necessary: one method to display the form, and a second method to analyse the content of the form and perform an adequate action such as doing a transaction on the database to update it. All this leads to very straightforward, but monotonous programming. The customising of Web pages probably needs some improvements so that this rather easy, but still time-consuming task, can be simplified.

One of the big advantages of the Web interface is that it is easy to sell to the users. Everybody knows how to navigate with a browser so that the look-and-feel is familiar. It is also easy to insert references to the database objects into other documents by simply grabbing the URL. One of our users started immediately to write scripts to get surveying information out of the database.

To resume, even if the Web interface is basically limited to one-level interaction, other advantages outweigh it when compared with X-based interfaces. With the WWW technology evolving rapidly and being integrated with other products it is also an interesting investment into the future.

3.3 Using O2 in controlling beam lines

Having implemented the BI database, the beam line configuration and the equipment information are readily available. Equipment is seen by experiments under different names (in fact an equipment such as a bending magnet can be seen as *BEND1* by one experiment and as *BEND3* by another one) which is easily solved by creating a named equipment reference. Beam lines are built from beam segments, which are already in the database.

What we need is an interface to the physical equipment to be able to read and set attributes such as magnet current or the position of collimator jaws. In fact there is no difficulty in calling equipment access functions, via the SL-EQUIP package[3], from equipment object methods. The corresponding O2 clients have to be linked with the SL-EQUIP library as it is shown in Figure 1.

The idea of equipment having properties (or attributes in other words), which can be read and/or set is a good paradigm for the equipment status and for the equipment control. The NODAL-based SPS control system used it extensively. The database is an excellent place to store some static attributes such as equipment settings and limits. Other attributes can change very frequently so that we have to access them directly from the equipment. In this case only the type and addressing information is stored in the database. A special case is when an attribute from the equipment has a setting and maybe tolerance as well, and it is often interesting to use them together. Finally the database can be used as a cache for attributes which are frequently used or expensive to obtain. A partial equipment class and equipment type class hierarchy is presented in Figure 2.

Offset corresponds to the offset in the attribute list and *dmProperty* is the Data Module property.

This definition can be used to produce a configurable, generic status display. Objects of class *EquipAttrType* can

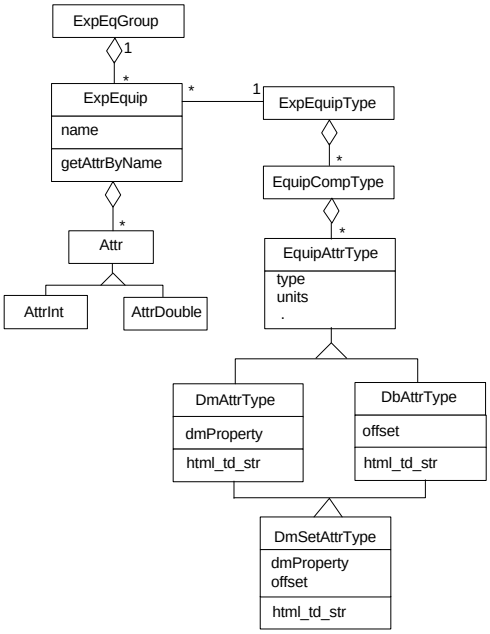

Figure 2

be added to the equipment type, describing where the equipment attribute has to be taken from. This approach has been implemented in our database. The method *html_td_set()* returns an HTML table cell so that status display in the form of a table can be generated.

3.4 Future work

The generic status display, which has been presented in the previous chapter, can be extended to setting equipment attributes to change current, position etc. There are no technical difficulties, but the problem of write permissions has to be carefully evaluated. The current permission schema is based on Unix user/group protection, which cannot be adopted. The HTML password schema currently used in our WWW interface has probably to be augmented with checking of the IP address of the request originator to make sure that the person setting attributes is actually on the CERN site.

Another typical operation for beam equipment is making a beam scan. This is a case where use of Java seems interesting given all the display facilities already available.

[1] An example page was foreseen in this place but it could not be produced on this space in an acceptable quality.

Scans could be stored in the database and compared with other scans.

As mentioned briefly before, the database could be used to cache equipment access. Performance determines whether it is interesting or not, so that detailed investigation of the response time and scalability are necessary.

To eradicate NODAL from our control system we will have to rewrite the access system, the surveillance programs and the distribution of the target data to experiments. In all cases the use of the new OO database seems to be promising.

4 Benefits from using object-oriented DBMS system

Programs written in an OO language such as C++ or Java often have to deal with persistent objects i.e. objects which remain the same beyond the lifetime of the program. In the accelerator area these objects can correspond to an abstraction of a physical component of the system such as a beam line or a beam device. Without an OO database these objects have to be explicitly created when the program starts or when it needs to use the object. For instance to find all collimators of the "H2" beam line we write:

```
d_OQL_Query q_eq(  "select e from b in
  TheBeamlineSegm, e in b->devices where
  b->name=$1 and e->type->name like $2)')
q_eq << " H2" <<   "Coll";
d_eql_execute(q_eq, eq_list);
```

This syntax complies with the ODMG C++ binding standard and is portable between different ODMG-compliant systems. The query statement in quotes is written in OQL query language, which is part of ODMG standard as well. All this leads to programs which are shorter, and more comprehensive than programs which use other types of permanent storage for objects. Another advantage is that such storage is flexible. When the class evolves, the program does not need to be changed in most cases. On the database level, any class of the schema can be easily modified without affecting objects of this class already stored in the database because new attributes will be initialised with default values for existing objects. In consequence software can be built in an incremental manner and this is exactly what we need to move from the old to the new control system.

Other benefits are the obvious benefits from using a commercial database system such as CASE tools for development, data security and optimisation of data access.

5 Conclusions

The new Object Oriented database has been a successful approach to the modernisation of the Experimental Areas software. The implementation of the BI database has allowed us to upgrade an important part of the NODAL-based system and introduce numerous improvements. More important, this approach is a good starting position to upgrade the whole system step-by-step. The use of an object-oriented DBMS and of the World-Wide-Web puts us in an excellent position to successively introduce object-oriented technology and Web tools. It is an innovative approach and could become a pilot project for other areas of controls in our group.

Acknowledgements

I would like to thank Dafydd Thomas for his help in importing the data to the database and understanding the equipment addressing. John Fullerton helped me to understand the old database and has provided most of the requirements.

References

[1] The New Controls Infrastructure for the SPS, M.J. Clayton and P. Charrue, Proceedings of the 1995 ICALEPCS conference, Chicago, 1995.

[2] PASS project at SSCL, PASS Note 93-1, also in IADBG Newsletter no. 9, http://www.cern.ch/IADBG /Welcome.html

[3] The Equipment Access Software for a Distributed UNIX-based Accelerator Control System, P. Charrue et al., ICALEPCS, Berlin, 1993