A NEW GENERATION OF SOFTWARE TO CONTROL SPECTROMETERS

Alain Buteau, Nathalie Ravenel, Katy Ho

The computing group of the LLB has been working for the last years on a new instrument control software called « Programme Orienté Objet de Pilotage de Spectromètres »

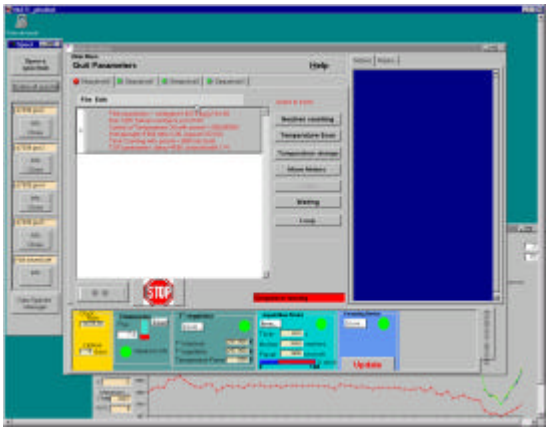

Figure 1. Graphical User Interface on spectrometer 7C2

The design guidelines

This new software generation has been designed with the following architectural guidelines :

- The software architecture should mimic the spectrometers architecture which are all different (and are all doing different physics studies) but have all many common points (electronic counting devices, sample environments, motors)
- Easy maintenance : The ability to provide a short response time to problems or required evolutions with a low manpower was a key point in the technical choices done for this software. The architecture must help the maintenance and evolution of the software. We reached this goal in describing the common points between spectrometers in high levels « modules » and let the differences be expressed in well isolated « modules »
- Extensibility: It should also be easy to integrate the control of new devices in the system to integrate as quickly as possible the experimental changes.
- Instrument driven : the complete instrument, experiment and environment description is done in configuration files. From these descriptions, the software adapts itself and constructs the corresponding Graphical User Interface and list of operations available on the spectrometer
- Friendly user interface : The experiment control software is the interface between the spectrometer and the physicist. Because the LLB has a lot of external beam users, a particular care should be brought to the design of the Graphical User Interface (see figure 1).
- Portability : the software must run on a Windows NT platform in a first time, but may also be used on Unix operating systems in the future.

The technical choices

To fullfill these requirements, the choice of Object Oriented (O-O) technologies seemed very natural. The O-O concepts of « modules » (called Class) and the inheritance and polymorphism mechanisms were potentially able to give us solutions to our problems. We decided to use the following techniques and tools to help us through the project.

1. An Object-Oriented notation : OMT and its successor UML has been used to help us during the analysis phase to :

- capture user requirements through Use Case and Sequence diagrams
- construct state models for objects with related life-cycles (example: Devices States)
- express the static structure of the system with Class Diagrams

1. An UML modelisation tool has been used to generate code skeletons from Class diagrams.
2. The « Design Pattern » techniques are used to bridge the gap between analysis and design. Patterns are an attempt to describe successful solutions to common software problems. They helped us construct a really modular software with documented solutions. In particular, it has allowed a clear separation between the Kernel part of the system in charge of the experiment control and the Graphical User Interface (GUI)
3. Portable and high level librairies : ILOG Views for the GUI construction, and STL for objects collection management.
4. Integration of the emerging Common Data Format of the Neutrons/Xrays community : NeXus.
5. And at least portable hand written C++ …..

And the (final) results

The software is now controlling 6 spectrometers of 3 different types :

- 1 Time of flight reflectometer : **EROS**
- 4 Two axis spectrometers : **G41, G61, 7C2, G52**
- 1 quasi-elastic time of flight spectrometer : **MIBEMOL**

The initial choices have proved to be adequate to our needs and are in the spirit of modern software engineering practices. The software is still evolving and the maintenance tasks have been contained at acceptable levels. We must nevertheless underline that the learning curve of Object Oriented technology has proved to be quite long and painful ! !

The future

The use of this software on the first 6 instruments since more than a year has proven the validity of the concept. Our goal now is to install the software on other spectrometers in order to check that it is able to be adapted easily on all type of spectrometers.

However, nothing is perfect and we are now working on the second generation software that will improve some weak points of the current version reported by the users. The data visualisation tools provided with the software are far from the standards met in common commercial program and should be updated. Another requirement is to provide a better remote access which will enable a easy and flexible remote control of the instrument. The use, in the next version, of a distributed architecture using CORBA as a software bus between the subsystems will enables us to provide this access through a Web Browser Graphical User Interface.