SMI++ - Object oriented framework for designing Control Systems for HEP experiments

C. Gaspar

CERN, European Organization for Nuclear Research
CH-1211 Geneva 23, Swizerland

B. Franek

Rutherford Appleton Laboratory, Chilton
GB-Didcot OX11 OQX

In order to cope with the complexity of the online control system the DELPHI experiment at CERN (Aarnio at al, 1991) developped, in collaboration with the CERN OC group, a new concept for the coding of the control logic. In this concept - SMI, the State Management Interface (Barlow at al, 1989)- the experiment is viewed as a collection of objects behaving as finite state machines. These objects are tipically organized in hierarchical structures allowing up to the full automation of the experiment by a a top-level object. This concept has been extended and is beeing re-designed using object-oriented techniques in SMI++ for the BaBar experiment at SLAC.

Key words: communication, control systems, development environments, object-oriented methods, programming languages

1 INTRODUCTION

The Online systems of physics experiments are normally composed of several different parts. Most common tasks are:

- The Slow Controls System (SC) controls and monitors slowly moving technical parameters and settings, like temperatures and high voltages of each sub-detector, and writes them onto a database.
- The Data Acquisition System (DAS) reads event data from the detector and writes it onto tape.

Preprint submitted to Elsevier Preprint *21 May 1997*

- The Trigger System provides the DAS system with the information on whether or not an event is interesting and should be written to tape.
- The Machine-Experiment Communication System controls the exchange of parameters between the accelerator control system and the experiment.

In previous experiments the control of the different areas was always designed separately by different experts, using different methodologies and tools resulting in a set of dedicated control systems.

DELPHI decided to take a common approach to the full "experiment control" (Adye at al, 1992) system. The result was the design of a system that can be used for the control and monitoring of all parts of the experiment, and consequently obtaining a system that is easier to operate, because it is homogeneous, and easier to maintain and upgrade.

2 THE SMI SYSTEM

SMI is a tool for developing control systems, it is based on the concept of Finite State Machines (FSM). Finite state machines are a simple way to describe control systems, complex systems can be broken down into small and simple FSMs that are hierarchically controlled by other FSMs. Using SMI the experiment can be decomposed and described in terms of objects behaving as finite state machines.

SMI objects can represent concrete entities, for example an hardware device or abstract entities like a logical sub-system. The objects representing concrete entities interact with the hardware they model and control through driver processes or proxies. The objects are typically organised in hierarchical structures called domains.

The interaction between objects can best be understood in the example of Figure 1.

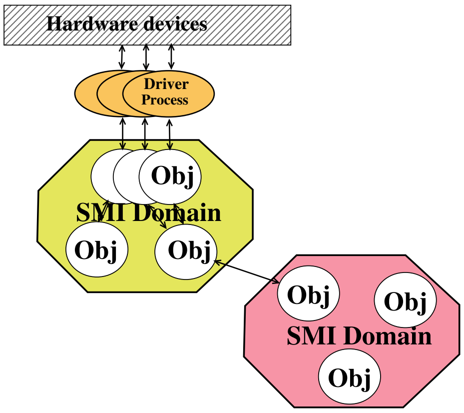

Figure 1. SMI example

The object model of the experiment is described using State Manager Language (SML). This language allows detailed specification of the objects such as their states, actions and associated conditions. The main characteristics of this language are :

• Finite State Logic

Objects are described as finite state machines. The only attribute of an object is its state. Commands sent to an object trigger actions that can bring about a change in its state.

• Sequencing

An action on an abstract object is specified by a sequence of instructions, mainly consisting on commands sent to other objects and logical tests on states of other objects. Actions on concrete objects are sent off as messages to the Driver Control Processes.

• Asynchrounous

Several actions may proceed in parallel: a command sent by object-A to object-B does not suspend the instruction sequence of object-A. Only a test by object-A on the state of object-B suspends the instruction sequence of object-A if object-B is still in transition.

• AI-like rules

Each object can specify logical conditions based on states of other objects. These when satisfied will trigger an action on the local object. This provides the mechanism for an object to respond to unsolicited state changes of its environment.

```
!- Example of SML code

object : RUN_CONTROL
  state : READY
    action : START_RUN
      do MOUNT TAPE
      if TAPE not in_state MOUNTED
        do MOUNT_ERROR ERROR_OBJ
        terminate_action/state=ERROR
      endif
      do START READOUT_CONTROLER
      if READOUT_CONTROLER in_state RUNNING
        terminate_action/state=RUN_IN_PROGRESS
      ...
  state : RUN_IN_PROGRESS
    when TAPE in_state FILE_FULL
      do PAUSE_RUN
```

```
    when READOUT_CONTROLER in_state ERROR
      do ABORT_RUN
    action : ABORT_RUN
      ...

object : READOUT_CONTROLER/driver
  state : READY
    action : START
    ...
  state : RUNNING
    action : PAUSE
    action : ABORT
    ...
```

The SMI mechanism allows an easy reconfiguration of the system: changes in the hardware can be easily integrated by modifying or replacing driver processes and logical modifications by changing the SMI code. The decoupling between the actual actions on the hardware (done by the Driver Processes) and the control logic (residing in the SMI objects) makes the evolution of a system from its first test phase up to final complexity a very smooth process.

3 SMI'S USE IN DELPHI

In DELPHI the full online system is controlled through this mechanism, the various areas of DELPHI have been mapped into SMI domains: sub-detector domains, DAS domain, SC domain, TRIGGER domain, etc. The full system comprises about 1000 SMI objects in 50 different domains.

A high level of automation of the experiment's control system is very important in order to avoid human mistakes and to speed up standard procedures.

Using the SMI mechanism the creation of a top level domain - BIG BROTHER - containing the logic allowing the interconection of the underlying domains (LEP, DAS, SC, etc.) was a very easy task.

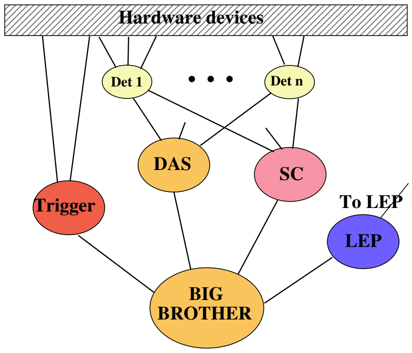

Figure 2. Big Brother Control

Under normal running conditions BIG BROTHER pilots the system with minimal operator intervention as shown in Figure 2. In other test and setup periods the operator becomes the top-level object, using the user-interfaces he can send commands to any SMI domain.

4 DISTRIBUTED ENVIRONMENTS

Current online control systems are generally characterized by a highly distributed architecture; like most computer control systems, they consist of workstations interconnected by a local area network.

SMI takes advantage of distribution, SMI Domains can run on a variety of computer platforms. The cooperation between SMI Domains including all exchanges between objects, are embedded in the SMI system. All issues related to distribution and heterogeneity of platforms are transparently handled by the underlying communication system -DIM (Distributed Information Management System) (Gaspar and Dönszelman, 1993).

DIM's aim is to provide interoperability between applications on different machines in heterogeneous distributed environments.

The DIM system was designed and implemented according to the following characteristics :

• Efficiency

The communication mechanism of DIM was chosen having in mind the asynchrounous character of SMI objects and the speed in reacting to changes or error conditions in the system. The solution we thought the best is for clients to declare interest in a service provided by a server only once (at startup), and get updates at regular time intervals or when the conditions change. I.e. an asynchronous communication mechanism allowing for task parallelism and multiple destination updates.

• Transparency

At run time no matter where a process runs, it is able to communicate with any other process in the system independently of where the processes are located. Processes can move freely from one machine to another and all communications are automatically reestablished. (this feature also allows for machine load balancing).

• Reliability and Rubustness

In an environment with many processes, processors and networks, it often happens that a process, a processor or a network link breakes down. The loss of one of these items should not perturbate the rest of the application. DIM provides an automatic recovery from crash situations or the migration of processes.

DIM uses a publish/subscribe mechanism. Any process in the Online System can publish (Server) information and any interface (or any other process) can subscribe (Client) to this information. A unit of information is called a "Service". A Name Server keeps track of all the Servers and Services available in the system.

Servers "publish" their Services by registering to the Name Server (Normally once at startup).

Clients "subscribe" to Services by asking the Name Server which Server provides the Service and then contacting directly the Server. Client's Services are then kept up-to-date in an event driven mode or at regular time intervals. Clients can also send commands to servers.

DIM is responsible for most of the communications inside the DELPHI Online System, it is used by SMI in order to transfer object states and commands, by the user interfaces in order to access SMI or any other necessary information and by many other processes for monitoring or processing activities. In the DELPHI environment it makes currently available around 25000 Services provided by 350 Servers. Dim is also beeing used by other experiments at CERN (and of course in Babar).

5 IMPLEMENTATION AND AVAILABILITY

DELPHI's SMI was implemented using ADA, each SMI domain corresponded to a file of SML code. The SML code was translated into ADA forming a VMS process, each object beeing mapped into an ADA task. This version is available on VMS only.

In SMI++ - The SML code is parsed and translated into an SMI object database that is then used by generic 'Logic Engines'.

The logic engine has been designed using an Object oriented design tool (Rational Rose/C++) and coded in C++ language. It uses the translated SML representation of the experiment to instantiate the required objects and then responds to external events to drive the computer model of the experiment.

Configuration tools allow the user to specify which objects belong to a specific SMI Domain, an SMI Domain corresponding to a Logic Engine.

The design of the project is completed and the implementation is in progress. The first prototype should be working by the end of the year.

SMI++ will be available on any mixed environments comprising : VMS (VAX and ALPHA) and UNIX flavours (OSF, AIX, HPUX, SunOS, Solaris)

DIM is already available in the above platforms and on OS9 and is beeing ported to WindowsNT and LINUX.

Other available tools are :

A generic SMI display, allowing to visualise the state of the SMI objects in a Domain and to send commands to them.

A DIM Display allowing the visualization of all the servers and clients in a certain DIM environment (including SMI and driver processes). Very usefull for debugging applications.

A DIM to WWW gateway, allowing access to all DIM services (including SMI states). The WWW page can be written in HTML with specific DIM tags containing the service name. The DIM tags are translated when the page is loaded.

6 CONCLUSIONS

SMI is a powerfull tool for designing and implementing control systems, it merges the concepts of object modeling and finite state machines.

The SMI system provides a simple language to model the application and a set of tools to compile, configure and run your applications on a variety of platforms.

The full control of the DELPHI experiment at CERN is implemented using this system, SMI proved capable of handling the control of different environments such as: data acquisition (including run control), slow controls, trigger, etc.

Due to the homogeneity in the control of the different parts of DELPHI it was possible to interconnect the different parts and completelly automate the DELPHI operations. It also considerably reduced the efforts on maintenance and upgrade of the complete control system of DELPHI.

SMI++ implements extensions to the SMI concept and is beein re-designed for use by the BaBar experiment at SLAC. The main extensions are related to more configuration capabilities at run-time, availability on a larger set of platforms (including heterogeneous distributed environments) and a higher support on graphical tools.

7 REFERENCES

DELPHI Collaboration, Aarnio, P. et al. (1991). The DELPHI Detector at LEP. In: *Nuclear Instruments and Methods in Physics Research A303*, pp. 233-276.

J. Barlow et al.(1989). Run Control in MODEL: The State Manager *IEEE trans.nucl.sci.36*, pp. 1549-1553.

T. Adye at al. (1992). The DELPHI Experiment Control *Proceedings of the International Conference on Computing in High Energy Physics '92.* Annecy, France.

Gaspar, C. and Dönszelmann, M. (1993). DIM - A Distributed Information Management System for the DELPHI experiment at CERN. In: *Proceedings of the IEEE Eight Conference REAL TIME '93 on Computer Applications in Nuclear, Particle and Plasma Physics.* Vancouver, Canada.