Data Handling Division
DD/87/24
October 1987

A KNOWLEDGE BASED WORK STATION TO CONTROL THE TRIGGER SYSTEM OF A HIGH ENERGY PHYSICS EXPERIMENT

G. Mornacchi and S. Vazzana

(Presented at the Europhysics conference on control systems for experimental physics, Villars, September 28-October 2, 1987)

DD-vf

A Knowledge Based Work Station to Control the Trigger System of a High Energy Physics Experiment.

G. Mornacchi, S. Vazzana
Data Handling Division, CERN
CH 1211 Geneva 23, Switzerland

Abstract

The drive towards higher complexity for High Energy Physics (HEP) experiments is a trend which is reflected into their component subsystems, e.g. the triggering hardware. Knowledge Engineering techniques offer a promising solution to the problem of designing and implementing software to operate complex systems. In particular model based reasoning techniques are also an attractive solution to the problem of integrating different control applications into a uniform system. In this paper we describe the approach taken to incrementally implement a work station intended to assist users of a complex trigger system during its design, test, operation and maintenance phases.

Introduction

he research activities in the field of Artificial Intelligence (AI) have concretised into a number of methodologies and techniques which provide new styles of programming and new ways of solving problems via computer programs [1].

Object Oriented programming (OOP) [2] and Knowledge-Based Systems (KBS) [3], the AI outcomes relevant to the work described in this paper, are in a mature enough state that powerful development tools are now available on the market [4], [5]. The application of these concepts to reasonable scale targets (e.g. a subsystem of an High Energy Physics (HEP) experiment) are therefore feasible. It is nevertheless our opinion that their application to a large scale case, such as an entire experiment, needs more results from the research.

Computer control is a field which has the potential to nefit from these new concepts. OOP and KBS are in fact widely applied to tasks such as fault diagnosis and monitoring in complex technical systems as well as to powerful Man-Machine Interfaces [1], [6]. In this context we are in the process of developing an application, targeted to an HEP experiment subsystem, which benefits from the use of OOP and KBS both from the problem solving point of view and (probably the most important aspect) on the software engineering aspects.

Application Area

The Energy Trigger system of the L3 experiment [7] follows the trends in size and complexity typical of the future generation of HEP experiments. It can be described very schematically as a complex interconnection of (roughly 100) modules, belonging to a small range of classes, such as ALU, memories, etc. The system is synchronous and driven by a central clock which distributes strobe signals to the modules according to a precomputed sequence. Data coming from the detector (the calorimeters) flow through the system which processes them and delivers a yes/no answer.

Development and Operation of such a system involve a number of tasks which can be broadly classified as:

- *Operator Interface:* such as initial set up, system configuration, interaction with other control tasks.
- *Monitoring:* assessment of the operating conditions of the system, including the detection of possible misbehavior (i.e. faults).
- *Troubleshooting:* operator assistance during the process of diagnosing a fault.
- *Engineering assistance:* assistance to the trigger developers, such as simulation, generation of the strobe sequences for the central clock.

Confronted with this list of tasks, our aim is twofold: solving the problems, where different tasks involve different problem solving techniques, and integrating the solutions into a coherent system with a powerful Man-Machine Interface. The current computer technology does offer us the hardware solution in the form of powerful Work Stations (WS) providing the necessary processing, memory and graphics resources, in addition to the networking capabilities to integrate the WS into the rest of the online system. The same, however, is not true on the software side. The traditional approach is that of developing separate computer programs (e.g in Fortran) loosely coupled through some sort of passive data base. While this approach lacks the necessary paradigms to tackle certain problems such as fault diagnosis, it also fails to produce a flexible and coherent system. Structured programming is the major software engineering concept, while issues

such as exploratory programming are not properly supported by the software development environments currently in use.

We believe that we need new software engineering concepts, both to optimize the development phase and to attack the difficult problems in an exploratory and incremental fashion, as well as new techniques to automatize a range of control problems, such as the use of rule based programming applied to fault diagnosis.

System Overview

Given the premises outlined in the previous section, we have found that results from AI research such as OOP and KBS provide us with a right set of concepts. A preliminary study [9], in fact, convinced us of the power of these concepts while also helping in the selection of a development toolkit which integrates these concepts into an industrialized product.

Our aim, translated into KBS terminology, is that of providing a range of Expert Systems[1] applied to the previously listed control tasks. We believe the concept of a Work Station, providing representation, reasoning and graphics capabilities, to be the right approach both to develop the problem solvers and to deliver an integrated product [10]. The key concept underlying this approach is that of model-based reasoning. A model of the problem domain is provided including the representation of the system (in terms of structure and behavior [8]) and its graphical presentation along with models and simulation for the relevant objects and concepts. This single base can then lead to several expert systems working in the same domain, where each expert system adds specialized knowledge (in a declarative or procedural form) to the base model. Their integration, as well as the Man-Machine Interface, is then achieved by exploiting the enriched base model.

Current Implementation

Following a preliminary work [9] on a Symbolics Lisp machine using its native Lisp with its Object Oriented extension [11], we have selected Intellicorp's Knowledge Engineering Environment (KEE) [4] as the software development toolkit. It still runs on a Lisp machine but an equivalent version for VAX/VMS stations is, by now, also available. The KEE toolkit includes the standard range of AI programming paradigms [12]:

- *Object Oriented Programming:* in the form of a frame based system with multiple inheritance which provides the necessary object modelling tools.
- *Access Oriented Programming:* in the form of "Active Values", frame slots with attached procedures which are fired whenever that slot is accessed.
- *Rule Based Programming:* in the form of a rules system providing the active use of declarative knowledge through the standard forward and backward inference mechanisms.

A powerful graphics subsystem integrated with the programming paradigms, access to the external world through the underlying Lisp language and a powerful tool to generate and maintain different views of the knowledge base are other tools available in the kit.

According to the ideas sketched above, we began by developing the base model of the trigger system in terms of:

- The prototypical description of the classes of objects in the trigger: modules, such as ALU, memories etc., buses and the central clock. The modules are described from the physical (position etc.) structural (connectors, registers, etc.) and behavioral (what the module does when activated) point of views. Lisp procedures (methods [4]) attached to an object description and accessed via message passing are used for the latter. Cables are represented as passive objects which memorize the signals flowing in the system between clock ticks. The clock distributes timing signals to the various modules by activating a method common to all module objects. To model the objects we have largely exploited the Object Oriented System to define a complex inheritance network which ends up into the description of the various classes.
- The model builder: a mouse driven facility to build and modify an interconnection of modules (i.e. a trigger circuit). In this way different circuits can be created interactively (and saved on disk to be used later on) while the system automatically handles the generation of the right object descriptions. Cables are automatically generated, if necessary, whenever two or more modules are interconnected.
- The model presentation: the objects modeled by the system have an associated graphical representation which is logically an integral part of the object description. A trigger model has associated a drawing into which modules and their interconnections (cables) are graphically presented. This graphical representation is active,

[1] Applications embodying expert knoweldge in a domain, such as diagnosis, along with the necessary capabilities to exploit this knowledge (inference or reasoning).

i.e. it both responds to mouse clicks (with menus of possible actions, such as inspection of an object (module, cable) description, insertion of probes) and can be animated by activating methods attached to the graphics units. In particular, the interactive interface to the model builder is built using the active object presentation: a module is selected from a menu of icons, it is positioned in the circuit drawing and it is then connected to the rest of the system.

Given this basic work station described above, we have then tackled the two tasks needed at this stage of the trigger "life": the simulation of a trigger circuit and the generation of the strobe sequences to be applied to the various modules in the circuit. The first one is fairly simple: the model itself can be run, and used as the simulator, by activating the module behavior under the control of the clock provided with the right strobe sequences. The second is more difficult because it depends on the structural knowledge held in the basic del (e.g. the relative position of the modules and their interconnections) as well as on knowledge which is local to particular instances of a module. This latter knowledge is typical of the task in question (generation of strobes) and can be related to the function of the module, i.e. what a module is used for when inserted at a particular position in the circuit.[2] The acquisition of this kind of knowledge is done directly by asking the expert who builds the model for additional information when a new module is created and entered in the system. In a sense the domain expert is expected to interact with the basic work station to "build the expert system". The generation of the strobe sequences is performed by simulating the circuit with a free clock (a clock distributing strobes to all modules all the times) while leaving to each module the task of deciding the presence/absence of the strobe signal for a clock tick. This decision is taken by exploiting both the "base" and the "specialised" knowledge.

Future

With respect to the extensions of the current system, we can list a number of short term aims:

- Porting from the development environment, now a specialized machine, to the target environment, a VAX/VMS work station. KEE does have both a development system, fully equivalent to the one running on Lisp machines, and a delivery system, stripped down of the development facilities, running on VAX machines. We are in the process of acquiring the VAX development version and we will do the port in the near future.
- Treating a new task: the testing of modules. Here part of the knowledge to be added to the base model will be in the form of existing test programs (e.g. written in Fortran). External programs can be easily integrated into KEE, by the external call facility found in the Lisp system, and it will be interesting to explore how this "non standard knowledge form" can be exploited (at least in our well delimited case).
- Interacting with a Data Base Management System. While the system knowledge base is in some sense a data base for the trigger, it cannot be completely decoupled from the rest of the experiment data base. In fact information needed to build the model (module positions, etc.) may already be available elsewhere, while the work station provides an easy to use interface to the information concerning the trigger. There are products catering for the interaction between an expert system toolkit and a DBMS, such as Intellicorp KEEConnection. We might be interested in trying them out.

Longer term aims involve the monitoring and diagnosis tasks. While it is early to have definite plans, we can anticipate that techniques such as rule based programming and causal reasoning could be the right way towards the development of these tasks [13].

Conclusions

We are developing a computer application to control and operate a complex trigger system, the design and the development of the application are based on methodologies, techniques and tools coming from AI research. In particular the concept of a knowledge based work station using model-based reasoning has been at the base of the approach. Software engineering methodologies such as those provided by OOP when coupled to a rich development environment supporting incremental programming have been a major factor behind the success of the project.

We are now at a stage where the basic work station including the domain description is ready and the two first application tasks are running on subcircuits of the entire trigger.

When assessing our experience we can reasonably claim that the approach taken has been successful for the design and development aspects of the project:

[2] For instance a particular ALU module should only sum over a well defined range of data signals, hence it should only be activated, i.e. receive strobe signals, when those data signals are present at its input connectors.

- The underlying concept of the knowledge based work station has been of great help in providing for a clean and flexible architecture. Application tasks share an active description of the domain and can be inserted incrementally into the sytem.
- The OOP concepts and the incremental development environment have been essential during the development process: the modelling of the domain objects, as well as coping with changes in their specifications, has been greatly easied, the exploration of different design and implementation approaches has been efficiently supported. The development time for the current implementation, not including the time needed to learn the tools and the design phase, can be evaluated to 3 man-months. We think this compares very favourably with different approaches.

Criticisms to this kind of approach are typically of two kinds:

- *Specialised languages and tools:* it is true that a toolkit such as KEE is complex, needs a certain investment to learn it and requires a minimal knowledge of Lisp. We believe, however, that, in addition to the direct outcome in terms of a product, the investment can pay off. Methodologies and techniques independent of the tool itself and applicable to other programming problems are acquired during the development. The same tool and the same ideas have a scope wider than the concrete application and could e.g. be applied to other parts of the experiment.
- *Performance:* languages like Lisp, hence tools built on top of it, have a bad reputation for performance when run on traditional hardware. While this has been true in the recent past, we think that the trend in computer technology is such that the necessary resources in terms of CPU power and memory size are already there in the latest generation of personal work stations (e.g. SUN-4, MicroVax 3000). These traditional work stations are, performancewise, competitive with specialized hardware while being less "exotic" and considerably cheaper. The same, however, cannot be said for the software: specialized Lisp machines are still well ahead in terms of programming environments and support for exploratory and incremental programming. The duality of "specialized" for development and "traditional" for delivery currently seems to be the best choice.

Acknowledgements

We acknowledge the support and the collaboration of the L3/University of Rome group, in particular R. Bizzarri and L. Luminari. We are grateful to our colleague F. Gagliardi for his successfull efforts to raise the interest at CERN in this new technology as well as for his major contribution in bringing at CERN the appropriate hardware and software tools.

We also wish thank M. Sendall, group leader of the CERN DD OC group, for his continous support and interest in our work.

References

[1] Proceedings "Conference on Artificial Intelligence Applications". IEEE Computer Society, 1985.

[2] B. J. Cox. "Object Oriented Programming. An evolutionary approach". Addison-Wesley, 1986.

[3] F. Hayes-Roth, ed. "Building Expert Systems". London: Addison-Wesley, 1983.
P. Jackson. "Introduction to Expert Systems". Wokingham: Addison-Wesley, 1986.

[4] Intellicorp, Inc. "KEE Software Development system User's manual". 1987.

[5] C. Williams. "ART Conceptual Overview". Inference Corporation, Inc., 1985.

[6] IEEE Computer. "Special Issue on Expert Systems in engineering". July 1986.

[7] L3-Rome Group. "L3 Level-1 Energy Trigger. L3J-1, 1987.

[8] R. Davis. "Diagnostic Reasoning based on Structure and Behavior". Artificial Intelligence, vol. 24, pp. 347-410, 1984.

[9] W. Fabbri, F. Gagliardi, G. Mornacchi. "Object Oriented description of the L3 Single Photon trigger System". CERN/DD/86/4, 1986.

[10] W. S. Faught. "Applications of AI in Engineering". In [6].

[11] Symbolics Inc. "Symbolics Lisp Reference Manual". 1986.

[12] D. G. Bobrow. "If Prolog is the Answer, What is the Question? or What it Takes to Support AI Programming Paradigms". IEEE Trans. on Soft. Eng., Vol SE-11, No. 11, pp. 1401-1408, 1985.

[13] F. Gagliardi, G. Mornacchi. "An Expert System Approach to computerized tools for online systems". Working Paper, CERN-DD, 1985.