Design Principles Employed in Aid of Real-time Expert Control System Development

W.F.S.Poehlman[*], *Wm.Garland*[**], *and J.W.Stark*[***]

ABSTRACT

In the approach to design knowledge-based control systems, certain fundamental principles can be applied to aid the coupling of slow knowledge transactions to high speed real-time data processing. Prime among these principles are the concepts of both temporal and functional abstractions.

For the enhancement of real-time performance within an intelligent control system and deliverability on a personal computer platform, a two-tiered multiprocessor architectural hardware structure has been adopted. At the lowest level, algorithmic procedures are vested in embedded controllers or single-board computers (SBC). At the highest level of control, an expert personal computer (PC) assimilates the monitoring information and responds with control directives that are carried out in detail at the SBC level. The latter processor uses traditional assembly language programming to accomplish time critical control tasks. The former processor operates more slowly but manipulates symbolically. Hence, the knowledge processor can be said to be operating with a higher level of temporal abstraction.

Software design includes the traditional modular development but is additionally clustered into functional layering schemes so as to conform to the hierarchical architectural platform on which the system is implemented. When viewed on a more global plane, such considerations prove to be examples of functional abstraction where information hiding at appropriate levels also aids real-time operation.

* Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, CA L8S 4K1.

** Department of Engineering Physics, McMaster University, Hamilton, Ontario, CA L8S 4M1.

*** McMaster Accelerator Laboratory, McMaster University, Hamilton, Ontario, CA L8S 4K1.

4th UNB AI Symposium
Fredericton, NB Canada
September 19-21, 1991

INTRODUCTION

The use of expert system techniques, to operate complex machines in a real-time regime, is in its infancy. However, the effort to develop high performance decision making knowledge-based systems is more than offset by advantages in their successful deployment. The slow response inherent in intelligent systems conflicts with the high throughput requirements of real-time systems. In the approach to implement such systems, certain fundamental design principles come to the fore which address this key issue that is required for successful expert system development. In this paper we explore these fundamental principles using a knowledge-based control system applied to the operation of a particle accelerator.

Design principles, for the purposes of this work, can be defined as application independent heuristics which address key issues by constructively constraining the system hardware and software realizations. The design approach followed below ranges from hardware configurations through to software design and is completed with the integration of both. Although centered around one application, the final sections serve to glean generic design principles.

The successful coupling of knowledge-based systems to real-time systems brings with it a myriad of advantages. Although those that follow have been derived from a particle accelerator application, most are generic in nature for process control regimes.

From previous work[1,2,3], the computerization of expert behaviour, when operating particle accelerators, has revealed several advantages. They include: (1) expertise on demand 24 hours a day; (2) an approach to process control which is always consistent, so that small deviations in machine behaviour (which indicate possible future failures or exhaustion of a consumable quantity) become evident in a much earlier time frame than when a human expert is involved (who usually unconsciously compensates); (3) focussing an operator's attention during times of process upset by tracking and filtering sensor data; (4) provision of on-line referencing from manual information; (5) generation of historical reviews where past operating histories are consulted and related to current operating scenarios; (6) diagnosis of abnormal states so as to generate recovery procedures; and (7) allowing the human expert more time to devote to other more complex duties or perhaps for instructional assistance to less skilled individuals, since tedious and routine control processes can be accomplished automatically by the expert system. Thus, the implementation of knowledge-based control systems, which requires special consideration in the form of extra design constraints, offers significant rewards.

BACKGROUND

The design strategies developed here have been applied to the development of an intelligent automated control system for a High Voltage Engineering Corporation Model KN particle accelerator within the Electronics Division of the Nuclear Effects Branch of the Defence Research Establishment Ottawa[4]. For this application, two areas were singled out as being most amenable to expert system approaches: (1) the human-machine interface and (2) direct computer control of primary operational parameters of the accelerator[5,6].

The human-machine interface found to be compatible with the majority of accelerator operators involves considerable graphical presentations. Figure 1 shows one representation developed as a graphical user interface (GUI). Although originally thought to be a study in intelligent interfacing employing user modelling techniques[7], this development proved to be adequately solved in real-time using a traditional programming implementation. The developed GUI, however, (implemented in Borland's Turbo PASCAL[8]) made extensive use of object-oriented features. From the developer's standpoint, the most useful aspect proved to be inheritance. From the user's point of view, "mouseable" operation for control was selected as the best interface attribute. Screen updates were timely and a complete screen refresh required much less than one second. It is ironic but understandable that the traditional operator control panel is analog (meter-based) in nature and the GUI final form replicates the analog signals on the terminal screen, after the computer monitoring system spends significant resources to digitize the signals.

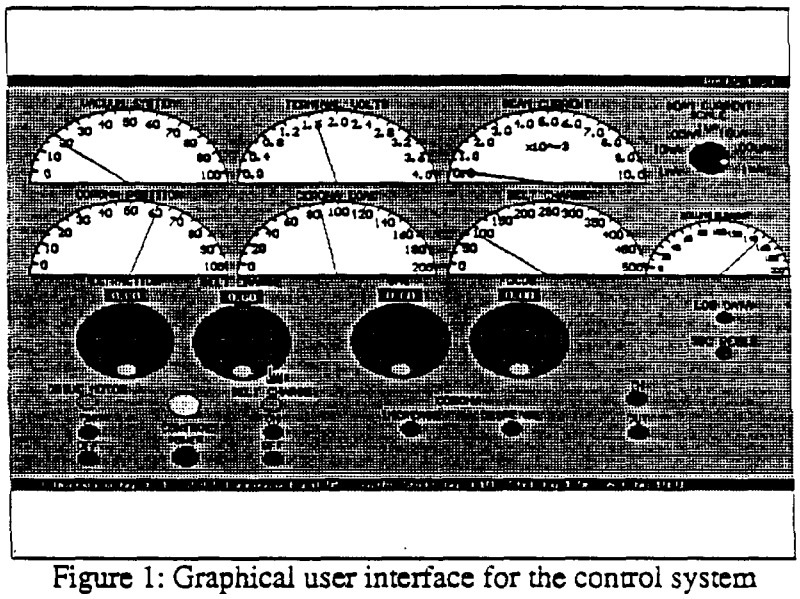

Figure 1: Graphical user interface for the control system

For this work, we shall be concerned with the direct control of system parameters (point 2 above) rather than the man-machine interface. As a control problem, accelerator control is not a major complex operation such as control of a nuclear power plant[9,10]. Having only four active control points, the maintenance of machine stability is tractable. However, there are 15 coupled and highly non-linear feedback parameters. Modeling accelerator operation analytically has never been adequately accomplished[11,12,13].

In analyzing accelerator system behaviour, three operating regimes occur naturally: (1) startup, (2) shutdown and (3) maintenance or ion beam stabilization and tuning. In all cases,

any non-nominal behaviour is detected and dealt with under a set of diagnostic procedures extracted from the expert leading to determinations of hardware failure, consumable exhaustion or unknowns. In the latter category, the orderly shutdown of the accelerator is the primary objective. Also, a continuous logging facility tracks machine directives as given by the expert controller and the corresponding accelerator response. Post mortem diagnoses form one method to check system operation and provides a mechanism for upgrades to the system.

DESIGN CONSTRAINTS

Early in the hardware design stage of the knowledged-based control system, it became clear that super- (better than) real-time capability is necessary in order to accommodate the hard real-time requirements[14]. However, inferencing is a notoriously computationally intensive operation; that is, not amenable to rapid real-time response. Thus, a parallel computational agent architecture is necessary, composed of both symbolic and numeric processors. The former perform higher-order knowledge-based decision making, while the latter perform data acquisition and direct control operations.

For the accelerator control system, certain software implementation considerations are evident. For example, there exists no model for machine upsets. Indeed normal operating parameters vary from day to day so that trend analysis/simulation has not proven useful. Only the experience of the expert machine operators has served as an adequate method for overall accelerator control that includes effective fault diagnosis and trouble-shooting. For instance, at the ion generation or source end, the physics of plasma behaviour is not well understood for either radio frequency or thermionic emission derivations. As for ion beam transport there are infinite values and combinations thereof for all the steerers, lenses and collimators that affect beam transmission. Hence, parameter uncertainty and irreproducibility preclude useful modelling of these accelerator characterisitics.

Therefore, for this application, pre-computed alternatives such as look-up tables, equations of state and/or decision trees do not provide the necessary capability for control. Look-up tables cannot provide sufficient resolution given resource requirements and required response time. Differential equations cannot yield a wide enough dynamic range unless directed by an intelligent agent thereby introducing undesirable multi-layer complexity. Decision trees become cumbersome for the depth required to cover all occurrences, and are extremely difficult to maintain and generalize. We turn, in summary then, towards symbolic processing and away from precomputation techniques.

In order to accommodate the global control modes of the application, certain characteristics must be present in the symbolic agent located at the topmost level of the control hierarchy. Specifically, during startup and shutdown, the system utilizes data-driven forward chaining to achieve the goal of a stable beam or deactivated accelerator, respectively. Fault

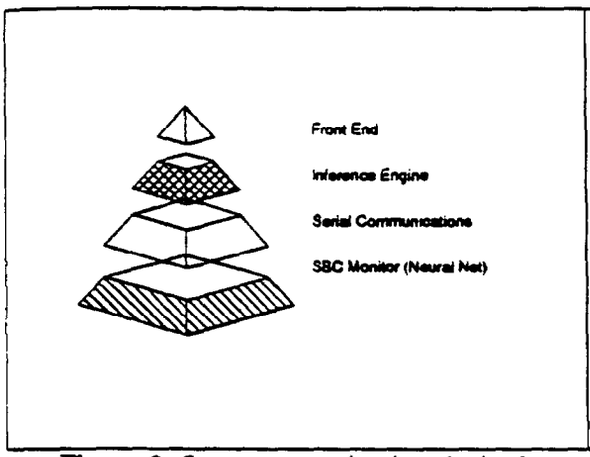

Figure 2: System organization desired.

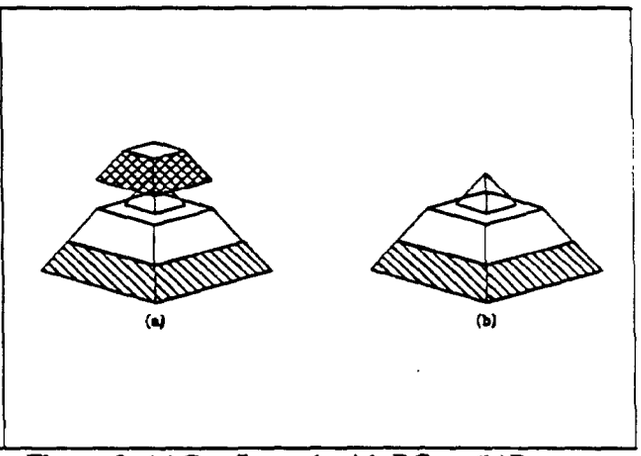

Figure 3: (a)Configured with PC+; (b)Program configured.

diagnosis requires goal-driven backward chaining to reach diagnostic conclusions based on accelerator performance data. Steady state operation involves both forward and backward chaining, depending on how the accelerator is performing and how the operator wishes to alter the accelerator's behaviour. Clearly, the control system must be able to switch freely between goal- and data-driven inferencing as events warrant. This requirement leads to the need for a flexible inference engine that is amenable to external control.

Hence, the selection of a PC-based shell for developing real-time expert system operation is critical. Rather than using a shell-based environment in which time-critical code must be cast, an embedded form of shell is much more flexible in providing real-time performance. On the other hand, the fastest method of delivering real-time decision making is to use so-called data-driven languages, such as ART-IM[15]. This, however, imposes constraints in which only forward inferencing is possible. This is not acceptable since diagnostic capability is best accomplished via backward chaining, which is also a requirement of the system.

The ideal two-tiered system architecture * is shown conceptually in figure 2. Originally, the controlling agent was envisioned as an expert system built around a Personal Consultant Plus[16] (PC+) knowledge base. It was found that PC+ is ill-suited for implementing rapid control directives due to: (1) its inability to operate as a subordinate component of a larger system; (2) its slow inferencing speed; and (3) its high demand for host system memory. Figure 3(a) shows how the tiers of the system are disrupted by PC+'s stubbornness to maintain top-level control.

---

* The reference to neural network techniques is under investigation at the SBC level. High level implementations show promise in developing self-organizing and adaptive expert controllers[17].

Another approach considered was to develop the system without using a pre-packaged expert system shell, as illustrated in figure 3(b). This path yields ultimate flexibility, but inference engine and knowledge base capabilities must be developed from scratch. As this is rather a "brute force" approach, it suffers from various drawbacks, such as: (1) slow development; (2) loss of maintainability and upgradability; (3) redundant re-implementation of existing software.

HARDWARE DESIGN

The final hardware configuration has taken the form of a two-tiered multiprocessor structure as shown in figure 4. At the lowest level, algorithmic procedures are vested in single-board computers (SBC). At the highest level of control and also at a higher level of temporal abstraction (that is, the information transfer is significantly more complex but the

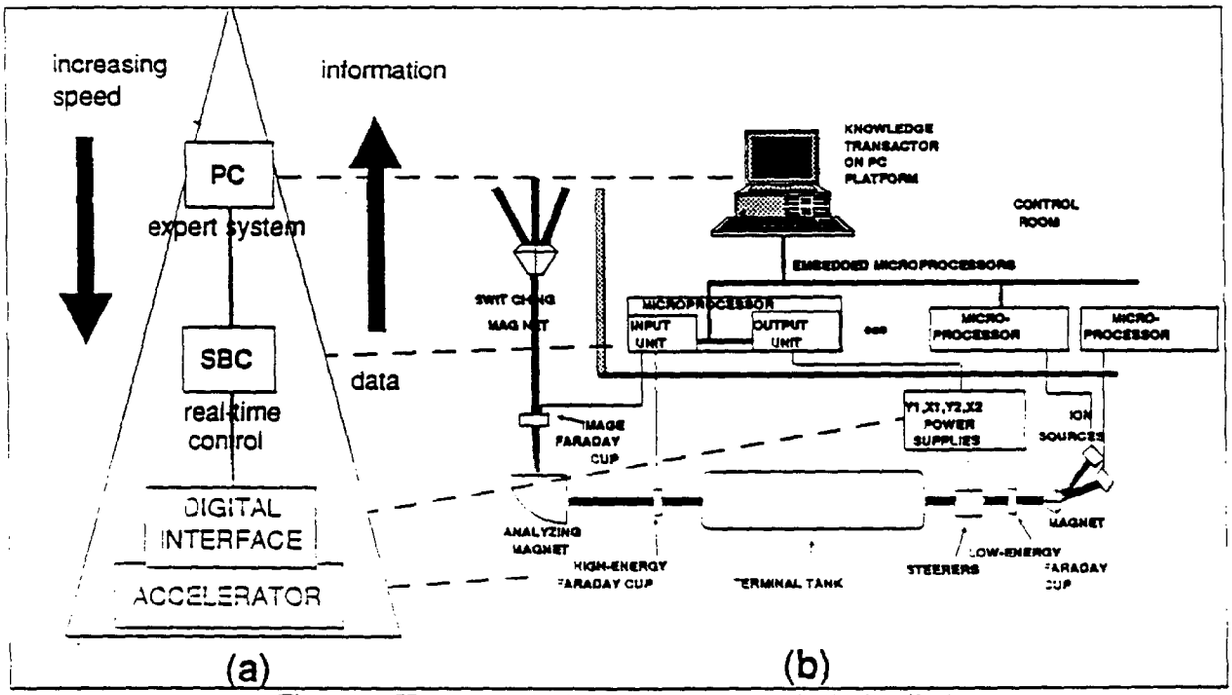

Figure 4: Hardware design -- (a)conceptual; (b)detailed.

transfer rate is of much lower frequency) an expert personal computer (PC) assimilates the monitoring information and reacts with control directives that are carried out in detail at the SBC level. The latter processor uses traditional assembly language programming to accomplish time critical control tasks. Although only one SBC is presently employed to operate the accelerator machine proper, expansion to a beam transport system controller via another SBC is underway. Capacity for high level direction of a larger number of SBCs by the PC is expected to be provided by hardware platform upgrade. (Eg. 386 processor to 486-based chips.)

SOFTWARE DESIGN

During the software design stage, allowance was made for the controller to exhibit certain modes of operation. Such planning accounts for the three naturally occurring operational regimes of the accelerator application: STARTUP, MAINTAIN and SHUTDOWN.

Under the STARTUP mode of operation, the purpose is to bring an ion beam of particular energy and intensity into a collision chamber. This involves ramping up currents and voltages of ion source components, charging high voltage terminals, and manipulating the resulting ion beam into the chamber. Once there and optimized, the beam must be MAINTAINED at steady state by frequent fine adjustments of control parameters. Finally, either at operator command or PC directive (when conditions warrant), the system SHUTDOWN procedure can be invoked. Such control segmentation reduces the scope of knowledge required thereby increasing response performance.

Also, built into each operating mode is a set of diagnostic heuristics which experts use to determine anomalous behaviour. Such heuristics have, as objectives, directives to either allow continued operation or invoke shutdown. Such a shutdown can be NORMAL (where, for instance, hot elements such as filaments are gradually cooled reducing thermal stress and increasing lifetime), RAPID (where minimum cooldown times are employed but the order of parameter down-ramping is adhered to) and PANIC (where all parameters are reduced at maximum slew rates). The latter mode will damage the machine in minor ways, but the alternative is catastrophic (and costly) failure if systems remain active.

Other software considerations include the implementation methodology for the symbolic processor. As mentioned above, the suitability of an embedded expert system shell is critical. For this system, Kappa-PC[18] (Version 1.0), a knowledge base development system and inference engine running under Microsoft Windows 3.0 was selected. Kappa-PC is designed to be extensible so that the programmer is able to tailor the expert system to suit the needs at hand. Although the bulk of the Kappa-PC package is distributed as compiled libraries to protect its proprietary nature, its main entry points are distributed as C source code along with instructions on how to customize the software. This property enables the system developer to embed the Kappa-PC system within a larger C computer program. Kappa-PC relies on the Microsoft Windows operating environment for interacting with the program user. This burdens the system developer with using both the Microsoft Windows Software Development Kit and Microsoft C compiler.

Kappa-PC is an object-oriented expert system shell that is based on the more complex KEE[19] system. Kappa-PC offers the following features:

- Hierarchical object-oriented knowledge base structure.

- High-resolution graphics user-interface platform.
- Fully controllable inference engine with forward and backward inferencing.
- Over 240 functions for inferencing, data manipulation, graphics, dynamic object creation, etc.

Thus, with all components selected or designed, the emphasis now is placed on system integration.

INTRA-SYSTEM DESIGN

The integration of the software/hardware configuration in the knowledge-based control system is shown in figure 5. Of critical importance is the interprocessor messaging system. That is, in the control plane for the accelerator system, the knowledge-based master communicates modes of operation (STARTUP / MAINTAIN / SHUTDOWN / DIAGNOSTICS), parameter setpoints and references, and goals to the embedded processors. The latter is particularly important during diagnostic phases of operation. In the reverse direction, embedded processors inform the master system of inability to comply and/or details of partial successes of compliance. Usually these messages take the form of accelerator machine behaviours which are non-nominal. Such situations lead to suspension of the current control goal and invocation of a diagnostician tuned to the problem at hand.

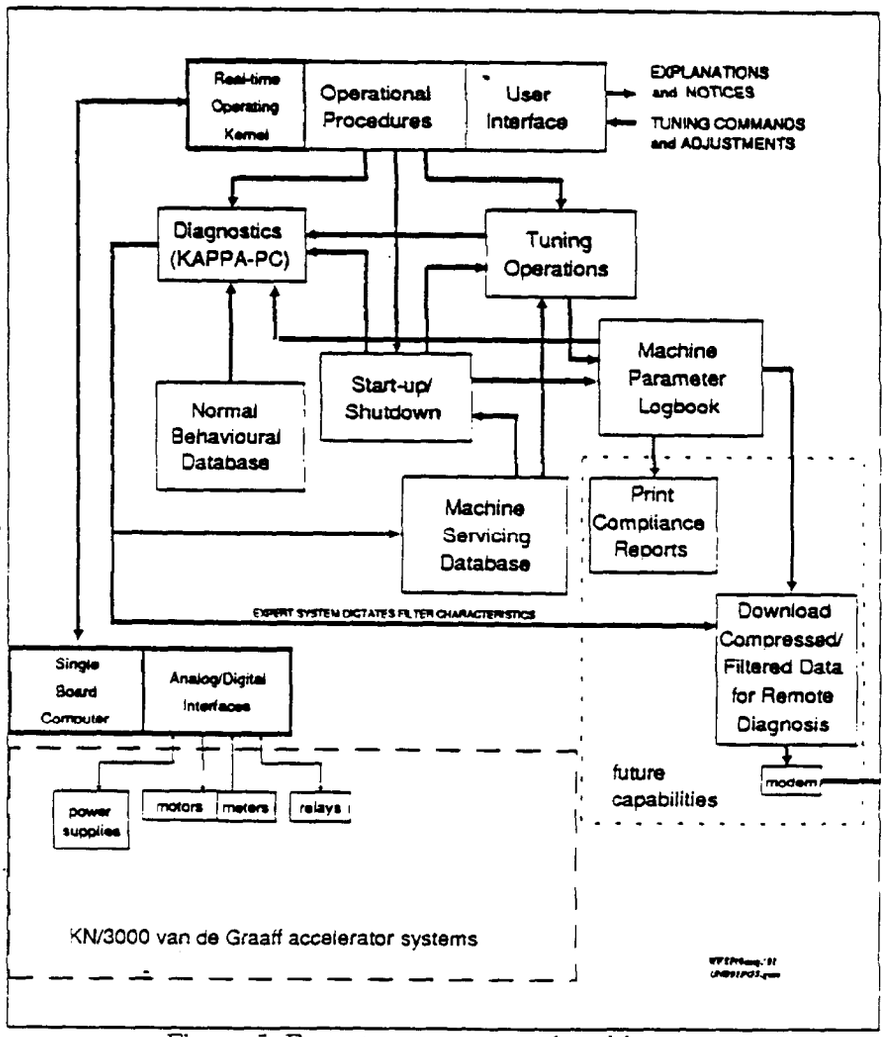

Figure 5: Expert system control architecture.

As an example, SBC-elicited communication packets need only be sent to the PC when the SBC detects data anomalies. There is only need for communications interplay when abnormal situations occur or new operating points are desired. New operating points are always accompanied by floor/ceiling windows which bracket acceptable behaviours for that parameter as determined by calculation (where possible) or by expert rule-of-thumb. These are generated at the upper level by the PC as required. Such reduced interplay aids in real-time performance. Simulated bench tests indicate that this control system is capable (under a nominal accelerator operating state) of traversing its control loop in several seconds which is an acceptable response time for stable accelerator operation.

DESIGN PRINCIPLES

The work undertaken here has revealed some principles that guide the decomposition of a system's functionality into modules or layers. Two primary layering techniques have been found to be extremely beneficial in defining the control structure: functional abstraction and temporal abstraction.

Functional abstraction follows from a natural logistical approach where information hiding is used to shield higher order levels of the system from actual specifics of the physical monitoring and control of the accelerator. At the topmost level, the system coordinator must manage:

- Communication with SBC: Initiation of commands to the SBC and reception of service requests from the SBC.
- User interaction: Compliance with user control actions received through the user-interface.
- Inferencing: Invocation of the inferencing sub-system to make decisions in response to either user commands or service requests from the SBC.
- Housekeeping: The logging of uploaded accelerator performance data and system integrity checking.

At the expert system level, responsibilities include

- Start up: Performs the start-up procedure.
- Shut down: Performs the shut-down procedure.
- Steady-state: Continuous fine-tuning of the accelerator to maximize beam stability while still affording the operator some capability to alter the accelerator's behaviour as required.

- Diagnosis: Evaluation of the accelerator's operating parameters in order to diagnose faults.

Thus the generic heuristic extracted here is to plan into the knowledge-based control system software design, functional layers tuned to the application's natural modes of operation.

Temporal abstraction is guided by incorporating a fixed partition strategy. Almost without exception, when control procedures could be made algorithmic, the process would be relegated to an embedded processor and implemented in assembly code and/or the "C" language. The single expert system handles heuristics and is charged with mimicking the expert accelerator operator using standard knowledge engineering and acquisition techniques. In this way the more time constrained modules operate at numeric processing speeds, while the less constrained modules operate with slower symbolic processing performance. Thus, the generic heuristic here is to partition knowledge-based control system software into modules which adhere to similar time constants and thereby create correlated temporal layers.

CONCLUSIONS

The development of multiprocessor control systems which marry the contained human expertise of symbolic processors with the speed of real-time systems involved in numeric processors allows the creation of intelligent resource management systems that directly and constructively affect the controlled environment.

Such complex system development is eased by adherence to certain fundamental design principles. Paramount among these principles for real-time knowledge-based control systems are abstractions of both functional and temporal natures. Guidelines which aid these structures include intelligent partitioning strategies and natural planning considerations that follow normal operating modes of the application.

REFERENCES

1. W.F.S.Poehlman and J.W.Stark, "Integrating Knowledge-based Systems into Operations at the McMaster Tandem Accelerator Laboratory", IEEE Trans.Nucl.Sci., *NS-36*, pp.1494-1498 (1989).

2. J.W.Stark and W.F.S.Poehlman, "Introducing Knowledge-based Control Systems into the McMaster Tandem Accelerator Laboratory: An Operations' Perspective", Nucl.Instr.&Meth., *A287*, pp.119-124 (1990).

3. D.Berg and W.F.S.Poehlman, "An Accelerator Operator's Companion for the McMaster University Model FN Tandem Accelerator", IEEE Trans.Nucl.Sci., *NS-36*, pp.1409-1417 (1989).

4. DND contract W7714-9-5971/01-SZ under the auspices of Dr. Tom Cousins, Scientific Authority.

5. P.C.Lind, W.F.S.Poehlman and J.W.Stark, "Implementation Considerations for PACES:the KN3000 Particle Accelerator Control Expert System", Proc.3rd Symposium/Workshop on Applications of Expert Systems in DND, May 2-3, 1991, at the Royal Military College, Kingston, Ontario pp.17-36.

6. S.DeMooy and W.F.S.Poehlman, "Using Expert Systems for Control Purposes", Proc.3rd Symposium/Workshop on Applications of Expert Systems in DND, May 2-3, 1991, at the Royal Military College, Kingston, Ontario pp.221-235.

7. R.Kass, "Implicit Acquisition of User Models in Cooperative Advisory Systems", Univ.of Penn., Report #MS-CIS-87-05/LINCLAB49 (Mar.,1987) 69p.

8. BORLAND, 1800 Green Hills Rd., P.O.Box 660001, Scotts Valley, CA 95067-0001.

9. W.J.Garland, W.F.S.Poehlman, N.Solntseff, J.Hoskins and L.Williams, "Intelligent Real-time System Management: Towards an Operator Companion for Nuclear Power Plants", Eng.Comp., *6* pp.97-115 (1989).

10. W.J.Garland and W.F.S.Poehlman, "The Use of Simulation in an Operator Companion for a Nuclear Power Plant", invited presentation for the AAAI'90 Workshop Programme -- Jul.29-Aug.3, 1990 at Boston, Mass., U.S.A.

11. P.H.Rose and H.Milde, "Transients in Large Tandem Accelerators", IEEE Trans.Nucl.Sci., *NS-18*, pp.63-67 (1971).

12. J.A.Staniforth and T.R.Charlesworth, "Spark Channel Characteristics of a Large Van de Graaff Generator", Nucl.Instr.&Meth., *188*, pp.483-489 (1981).

13. J.A.Staniforth, "Voltage Calculations for a Single-ended Van de Graaff Machine with Secondary Breakdowns", Nucl.Instr.&Meth., *216*, pp.1-9 (1983).

14. A.Mok, Fundamental Design Problems of Distributed Systems for the Hard-real-time Environment, Ph.D. Thesis, Massachusetts Institute of Technology, Cambridge, MA., 183pp. (May,1983).

15. Inference Corporation, Los Angeles, CA.

16. Personal Consultant Plus, copyright Texas Instruments Inc., Austin, TX., USA.

17. C.G.Looney, "Rule Acquiring Expert Controllers", IEEE Trans.Know. & Data Engg., *3#2*, pp.252-256 (1991).

18. KAPPA PC copyright Intellicorp, Mountainview, CA., USA.

19. Knowledge Engineering Environment, copyright Intellicorp, Mountainview, CA., USA.