CV ref P41

COVER SHEET

THE DESIGN AND IMPLEMENTATION OF AN OPERATOR'S PERFORMANCE SUPPORT SYSTEM

by

R. J. Wilson.
EACS - Engineering and Computing Services,
82 Rayne Avenue, Oakville, Ontario,
Canada, L6H 1C2

A. A. Bokhari.
Department of Computer Science and Systems
McMaster University, Hamilton, Ontario,
Canada, L8S 4K1

Wm. J. Garland.
Department of Engineering Physics
McMaster University, Hamilton, Ontario,
Canada, L8S 4L7

W.F.S. Poehlman.
Department of Computer Science and Systems
McMaster University, Hamilton, Ontario,
Canada, L8S 4K1

and
C. W. Baetsen.
Department of Engineering Physics
McMaster University, Hamilton, Ontario,
Canada, L8S 4L7

Paper Log: 092
June 1993

Camera ready summary for publication in the Book of Summaries and camera ready paper for publication in the Proceedings of the International Nuclear Congress '93, October 3-6, 1993, Toronto, Ontario.

THE DESIGN AND IMPLEMENTATION OF AN OPERATOR'S PERFORMANCE SUPPORT SYSTEM

by

R. J. Wilson, EACS - Engineering and Computing Services, 82 Rayne Avenue, Oakville, Ontario, Canada, L6H 1C2

A. A. Bokhari, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada, L8S 4K1

Wm. J. Garland, Department of Engineering Physics, McMaster University, Hamilton, Ontario, Canada, L8S 4L7

W.F.S. Poehlman, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada, L8S 4K1
and
C. W. Baetsen, Department of Engineering Physics, McMaster University, Hamilton, Ontario, Canada, L8S 4L7

Summary

The objective of the current OPUS (Operator/User Support) System project is to produce an intelligent performance support system for the operators and maintainers of the Central Sampling and Condenser Leak Detection System at the Pt. Lepreau Nuclear Generating Station in New Brunswick. The design of the OPUS system addresses issues that have been raised in recent reviews and takes advantage of the experience that has been gained in the area of operator support systems over recent years. The fundamental design revolves around a user-centred concept since this will help to address one of the major shortcomings of past designs which is lack of operator acceptance. OPUS is also an active system in that it is designed to be responsive to changing circumstances. The model will eventually run on different platforms, for both performance and geographical reasons, and is multi-tasking in order to realize the inherent parallelism that is present. Each task is a separate software process and, to allow for incremental growth, proactive, persistent and intelligent modules are used. This is singularly useful in lessening the software maintenance requirements. An anthropomorphic three level (strategic, tactical and operational) model has been adopted that utilises the blackboard architecture. The consequential and concomitant aspects of communication, synchronisation, and organization of the computational flow for real-time performance create problems of their own, not least of which is the intertwined effects of temporal and logical correctness. The communications traffic is handled both asynchronously and synchronously and decomposed into four distinct categories. In order to develop the concepts on a real problem, the system for sampling the secondary side chemistry at the Pt. Lepreau Nuclear Generating Station was chosen. This phase is a single processor realisation but it is expected to rapidly expand into a multi-processor multi-platform implementation encompassing a number of operating systems.

THE DESIGN AND IMPLEMENTATION OF AN OPERATOR'S PERFORMANCE SUPPORT SYSTEM

by

R. J. Wilson, EACS - Engineering and Computing Services, 82 Rayne Avenue, Oakville, Ontario, Canada, L6H 1C2

A. A. Bokhari, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada, L8S 4K1

Wm. J. Garland, Department of Engineering Physics, McMaster University, Hamilton, Ontario, Canada, L8S 4L7

W.F.S. Poehlman, Department of Computer Science and Systems, McMaster University, Hamilton, Ontario, Canada, L8S 4K1
and
C. W. Baetsen, Department of Engineering Physics, McMaster University, Hamilton, Ontario, Canada, L8S 4L7

ABSTRACT

The objective of the current OPUS (Operator/User Support) System project is to produce an intelligent performance support system for the operators and maintainers of the Central Sampling and Condenser Leak Detection System at the Pt. Lepreau Nuclear Generating Station in New Brunswick. The design of the OPUS system addresses issues that have been raised in recent reviews and takes advantage of the experience that has been gained in the area of operator support systems over recent years.[1,2] The fundamental design revolves around a user-centred concept since this will help to address one of the major shortcomings of past designs which is lack of operator acceptance. Primarily the system is an active one in the sense that it reacts to changes in input data and undertakes circumstance sensitive tasks. The model will eventually run on different platforms, for both performance and geographical reasons, and is multi-tasking in order to realize the inherent parallelism that is present.
The contrived modularity is extremely germane to minimising code maintenance requirements. Additionally it eases the task of code verification and provides for incremental growth and flexibility. Asychronous communications are used to provide flexibility and to handle the parallelism while to ensure stability periodic synchronous communication is imposed.

INTRODUCTION

The OPUS (Operator/User Performance Support) system is a group of interrelated computer programs designed to assist in decision making and to monitor data on a continuing basis for a very specific set of preenumerated procedures. An anthropomorphic approach has been taken for two major reasons. These are to improve operator acceptance and because the problem breaks down into naturally into three layer strategic, tactical and operational aspects.

We have named these the MANGER, SUPERVISOR, AGENT levels. In addition it is designed to act as a test bed for a range of generic ideas.[3] The model will eventually run on different computing platforms, for both performance and geographical reasons, and is multi-tasking in order to deal with the concurrent requirements of the problem. Each task is a separate software process and, to allow for incremental growth, proactive, persistent and intelligent modules are used. The consequential and concomitant aspects of communication, synchronisation, and organization of the computational flow for real-time performance create problems of their own, not least of which is the intertwined effects of temporal and logical correctness. In order to address some of these, all communication is effected through the blackboard. The communications traffic is handled both asynchronously and synchronously and decomposed into four distinct categories.

To develop the concepts on a real problem, the system for sampling the secondary side chemistry at the Pt. Lepreau Nuclear Generating Station in New Brunswick was chosen. The current phase is a single processor realisation but it is expected to rapidly expand into a multi-processor multi-platform implementation encompassing a number of operating systems. The procedures used to define the code within the OPUS modules are found in internal documents from the New Brunswick Electric Power Commission - Pt. Lepreau. It became apparent early on that the nature of the problem set required parallel processing and the entry of data at distributed locations.

Problem Description

In order to fully define the practical problem as far as the nuclear plant was concerned a number of documents and flow sheets for the Pt. Lepreau secondary side chemistry were consulted. These outlined procedures for dealing with the problem of bad chemistry brought about by either an ingress of cooling water, due to a condenser leak, into the system or from some other cause. The procedures for a condenser leak as opposed to bad chemistry are sufficiently different to allow the problem to be

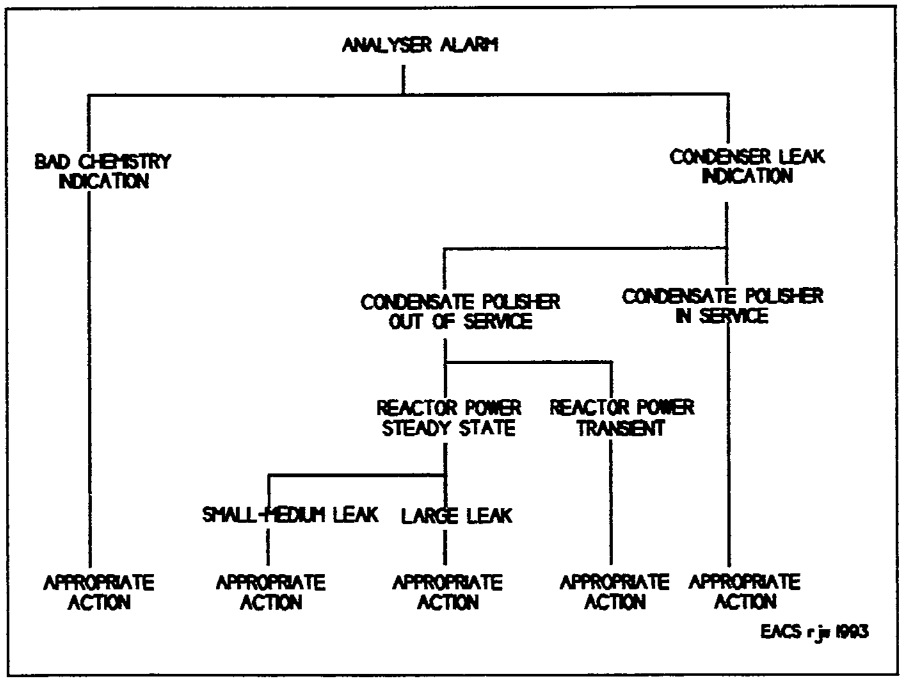

Figure 1 **The Problem Logic**

broken down on this basis. Figure 1 shows the logic involved. It may be seen that if bad chemistry is discovered the procedures to be followed by the maintainers is the same regardless of the reactor power level and whether or not condensate polishing is on or off. On the other hand if a condenser leak occurs a more complex set of decision making logic is involved and the specific criteria for actions to be taken depend on the reactor power level and whether the plant is undergoing startup and whether condensate polishing is on or not. This logic is spread out over a series of individual programs that are called into action only when required by the current situation. The secondary side system has monitoring points where the maintainers are required to take manual grab samples both to confirm a bad chemistry or condenser leak situation and to monitor the situation as the event continues. Trying to execute the process in a serial fashion is impracticable because such a programme would have to wait for the input of a grab sample. These are required only each 30 minutes so that response to other events would be prevented. This would defeat the whole purpose. This inherent parallelism in the problem requires that event driven programming techniques be used.

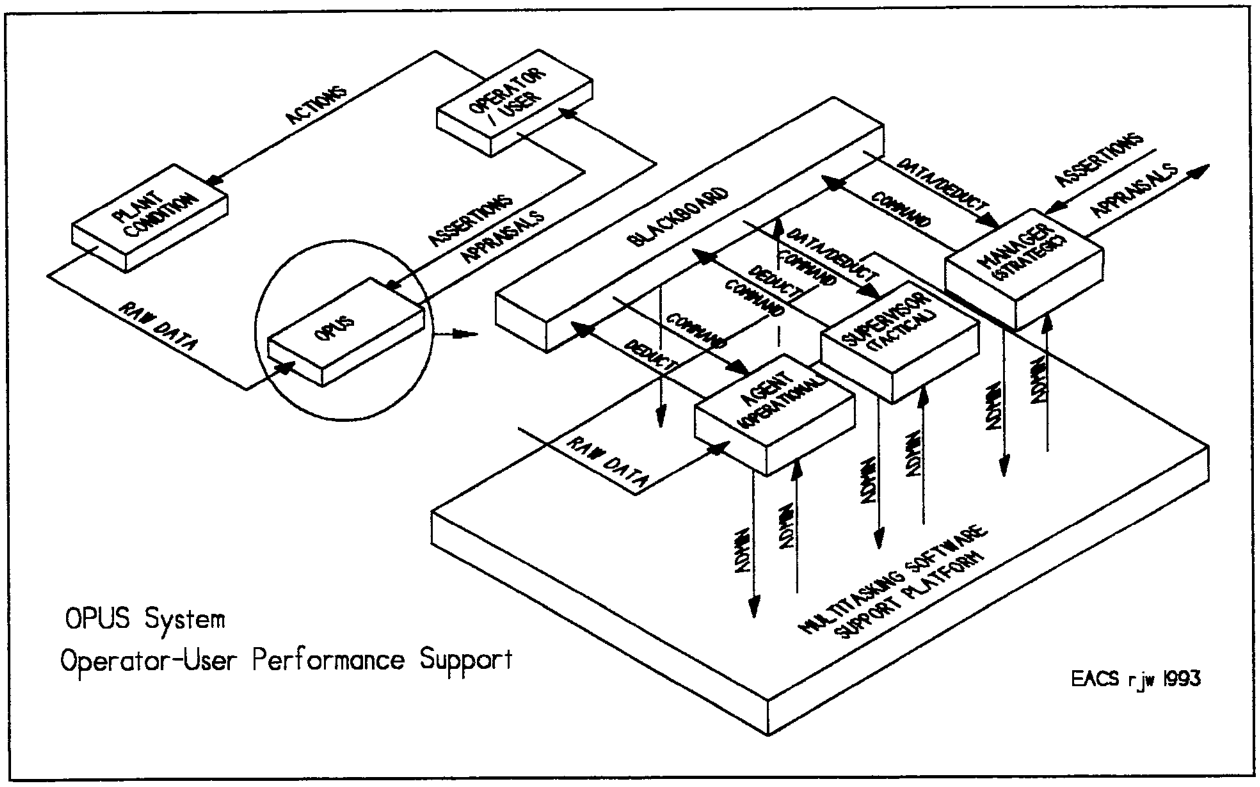

Figure 2 The OPUS Multitasking Model

The OPUS System Model

For the foregoing reasons and for others that will be discussed the solution was recast in multiplatform/multitasking terms as illustrated in figure 2. This shows how the user interacts with OPUS. The OPUS system accepts raw plant data and supplies its appraisal of the current situation to the user. The user in turn can request more information and investigate the reasoning behind this assessment until satisfied. This does imply that the user have a very clear understanding of the OPUS system just as he does of the existing operating manuals. OPUS does not filter data. Rather it is a reactive system which presents the user with the circumstance sensitive information and reasoning processes it uses in a layered and interactive manner. A pointed out by Garland[4] the mental model of the plant will differ depending on whether an operator, engineer or technician is using is the system. OPUS accommodates this by giving the user control of how much or how little information is to be displayed. A control room operator for example may only require summary messages of what the current situation is and whether procedures are being followed to deal with it. A chemical maintainer on the other hand will require more detailed information on what steps are required to be taken and should be prompted to enter data.

- **NO ALARM**
- **ONE LEAK ALARM**
- **UNRECOGNISED ALARM**
- **LEAK ASSUMED**
- **BAD CHEMISTRY**
- **LEAK & BAD CHEMISTRY**

Figure 3 Alarm Handling Agent. DEDUCTIONS

The hierarchical anthropomorphic design has essentially three levels:(1) strategic, which we refer to as the MANGER level, (2) tactical, which we refer to as the SUPERVISOR level and (3) operational which we call the AGENT level. In accordance with the user-centred approach discussed by Garland et al[5] the users of the system can interact with the different levels depending on their needs. The core of the program revolves around a central blackboard whose purpose is to coordinate the inter-module communication and supply data as required to each of the system modules.

Agents have very specific and limited tasks to do. They only accept input data from the plant and commands from their supervisors. Normally they do not communicate with other agents or other supervisors. They do not take actions by themselves, they merely relay conclusions back to their supervisors and display them locally as requested by the user. An example of these conclusions is shown in figure 3 for the Alarm Handling agent.

The role of the Supervisor is more complex in that, in addition to dealing with the same inputs as the agents, they must also deal with the deductions of the agents they are supervising. The Supervisor also takes no actions itself but communicates its own assessment of the situation to the Manager. In some instances this may simply be passing on the messages from its agents.

The Manager accepts as input all the deductions from its supervisors, keeps the user informed, accepts user commands, monitors problem solving resources, spawns or kills off tasks as required and issues commands to its own supervisors. It is the only module that takes actions on its own initiative since it can spawn other processes as and when needed.

The code used to construct the model is generic, only the procedural contents of the modules are problem specific. This may be likened to using building blocks where the nature of the code blocks may be changed without effecting other modules that are on the same level or below them. This imparts a number of features

- modules can be run on the computing platform of choice depending on the geographical location and/or computational requirements and tasks can be dealt with in parallel.
- the procedural rules may be changed within a module without effecting the others.
- the user has control in the sense that a different module(s) for different situations can be used. For that matter, modules with differing reasoning procedures may be used for the same situation if they are available.
- modules are only run on an "as required" basis, thereby conserving computing resources.

The Blackboard
One of the major basic components of the overall structure is the blackboard. Essentially, the blackboard consists of a block of global memory which is shared by a number of programs. These programs can write to or read from the blackboard and may sometimes consist of separate segments of knowledge called the knowledge sources. The blackboard provides a loose coupling which makes the modules more autonomous. In theory the design allows for a large enough number of modules (limited only by memory) to cover most situations. This large number need not slow the completion of tasks because agents are only invoked when they are needed and the use of a distributed architecture allows extra processes to be added as necessary to accommodate more agents.

All communication between various modules is through the blackboard and can be broadly divided into two types. The messages from individual modules to the blackboard and the messages from the blackboard to various modules. , the first kind of messages are received asynchronously by the blackboard. This asynchronism aids exploitation of any parallism that may be present within the problem and provides the flexibility. However despite the desirability of completely asynchronous behaviour, both Burke and Prosser[6] and Benveniste[7] point out that it is undisciplined, potentially unstable and unpredictable. Thus the management strategy of imposing sychronous behaviour is employed on the output side of the blackboard to direct the problem solving effort. This ensures that all modules are processing the same set of data and have taken into consideration any results obtained by processing of the previous data set. However this does imply that agents finish computation before the new data set arrives. The asychronous input messages are saved by the blackboard and are transmitted synchronously along with a new set of data to all the modules when the new data set becomes available. The messages for an individual module are placed in a first-in-first-out queue and are read by the module sequentially.

Communications Types

Clearly, on a fully distributed system, communication will take place over a network using the tools the operating system(s) provide. However the message traffic between modules and can be decomposed quite naturally into one of four fundamental generic formats and it is only the mode rather than the content of the messages that will change. These formats are labelled , DATA, COMMAND, ADMINISTRATION and DEDUCTIONS.

The DATA messages contain raw input data from the plant, that is instrument output and plant status data such as reactor power and other operational data. The COMMAND messages contain the instructions from higher level modules to lower ones regarding use of resources and operating modes. The ADMINISTRATION messages contain such things as system time. The DEDUCTIONS messages contain the conclusions that each of the modules arrives at based on the current operating status, raw plant data and, if appropriate, deductions from other modules.

Figure 4 shows the modules that are required for monitoring the situation under normal operation. Each process , a Supervisor and its Agents, is concerned only with a specific set of conditions and in fact does not know how to deal with events outside its own realm of expertise. For example the Data Monitoring Supervisor in Figure 4, is only required to deal with instrument maintenance, monitoring for abnormal trends (alarms) and verifying that the status

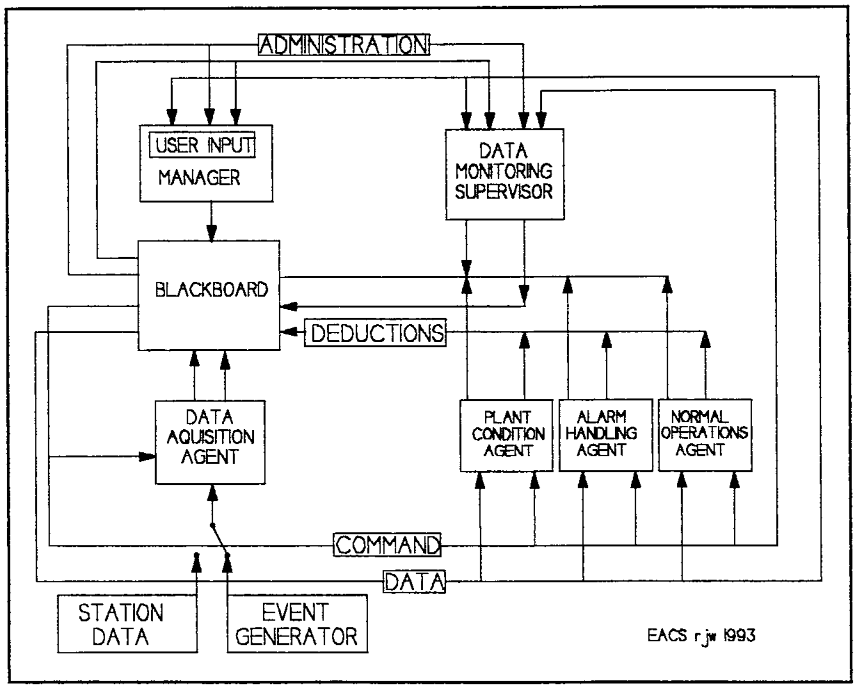

Figure 4 Multi-Tasking Model. Normal mode

of OPUS and the status of the plant is rational. Thus we have the Normal Operations agent, the Alarm Handling agent and the Plant Condition agent all reporting to the Data Monitoring supervisor via the blackboard. Should a Chemical upset occur, ie bad chemistry or condenser leak, this information is passed onto the Data Monitoring Supervisor which would inform the Manager. It is the task of the Manager to spawn the resources that are required for the specific event and to keep the user informed of its actions and the reasons why.

Event Generator

Since actual plant data was unavailable during the development stages of OPUS it was necessary to construct a means for testing. The Event Generator, shown in figure 4, was built for this purpose. The Event Generator programme consists of three basic sections: direct alarm toggling, simple scenario and dynamic plant simulation. The programme is menu driven and it allows the user to make a series of choices that are immediately available to the OPUS system.

The alarm toggling section allows the user to toggle any alarm or combination of alarms on or off. This allows verification to be performed on the logic incorporated into the various OPUS modules that depends on analyser alarm settings. The simple scenario section provides a selection of scenarios for a number of monitoring points. This allows verification of the logic that depends on monitoring the trend from the Central Sampling System analysers. This is somewhat limited in scope because only one analyser trend was used as a trigger for each event and trends were not based on real physical processes. The dynamic plant simulation section will be designed to overcome the above limitations in that it is intended to simulate a condenser leak or bad chemistry scenario based on the behavioral characteristics of the secondary side system itself. In particular this should prove valuable as a training platform for operators and maintainers.

Current version of the OPUS

The present version of OPUS is a single processor model running under DesQview/X layered on MS-DOS. It is currently undergoing evaluation at the Pt. Lepreau nuclear generating station. In addition work is proceeding to expand the system into a multi-processor multi-platform implementation encompassing a number of operating systems using the X-Windows open standard protocol.

Conclusions

The major issues that have been raised in contemporary assessments of operator support systems have to a large extent been addressed in the design of the OPUS system These are a user-centred concept to aid operator acceptance, different platforms for performance and geographical reasons, multi-tasking to handle the parallel nature of the problem, modularity to assist verification and code maintenance, and provide incremental growth and flexibility, asychronous communications to provide flexibility and handle parallelism and synchronous communication to ensure stability.

The OPUS system is currently under evaluation by the maintainers and operators of the Central Sampling and Condenser Leak Detection System at Pt. Lepreau Nuclear Generating Station in New Brunswick. Use of the system at Pt. Lepreau will materially effect the ultimate conclusion with regard to the one outstanding issue, that is of operator acceptance.

Acknowledgements
The authors would like to express their appreciation for the encouragement and cooperation of the staff at the Pt. Lepreau nuclear generating station in New Brunswick and for the financial support of the Natural Sciences and Engineering Research Council of Canada under the Strategic Grants Program, award number STR0118177.

References

[1] Bernard, John A., "Issues Regarding the Design and Acceptance of Intelligent Support Systems for Reactor Operators", IEEE Trans.Nucl.Sci., p.1549-1558(1992).

[2] Garland, Wm. J., and Poehlman, W.F.S., " Impact of Operator Aids on Human Factors and Organizational Structure", AECB(Atomic Energy Control Board of Canada) report, (to be published)

[3] Garland, W.J., Poehlman, W.F.S., Wilson, R.J. and Bokhari, A.A. "Towards a Generic User Support System(GUS)", Canadian Nuclear Society Fourth International Conference on Simulation Methods in Nuclear Engineering, June 2-4, 1993, Montreal, Quebec, Canada.

[4] Garland, Wm.J., "Dealing with Disparate Mental Models", accepted for presentation at HCI International '93, Orlando, Florida, USA, August, 8-13, 1993.

[5] Poehlman, W.F.S., Garland, W.J., Bokhari, A.A., Wilson, R.J. and Baetsen, C.W., "Performance Support Systems and Artificial Intelligent Considerations", INC93 -- International Nuclear Congress 1993 , October 3-7, 1993, Toronto, Canada

[6] Burke, P., and Prosser, P., "A Distributed Asynchronous System for Predictive and Reactive Scheduling", Artificial Intelligence in Engineering, 1992, Vol.6(12), pp106-124.

[7] Benveniste, A. and Berry, G., "The Synchronous Approach to Reactive and Real-Time Systems", Proc. IEEE, Vol.79,No.9, September 1991.