INTELLIGENT CONTROL AND COMMUNICATION SYSTEMS

L.C. Schooley, F.E. Cellier, F.-Y. Wang, and B.P. Zeigler

Department of Electrical and Computer Engineering
and
Department of Systems and Industrial Engineering

University of Arizona

Abstract

We describe progress on the development of a high-autonomy intelligent command and control architecture for unmanned plants that conduct scientific experiments or process local planetary resources. Examples include a science rover suitable for lunar or martian deployment and an automated process plant for production of oxygen from carbon dioxide. A distributed command and control architecture has been designed to teleoperate such plants in a high-level task oriented mode with supervisory control from one or several remote sites. The architecture integrates advanced network communication concepts and modern man/machine interfaces with recent advances in autonomous intelligent control. A complete working prototype is being developed to demonstrate the architecture. Experience with, and lessons learned from, this development project are reported.

INTRODUCTION

A distributed command and control architecture is being developed that ultimately will provide the capability to teleoperate one or several scientific experiments or materials processing plants located on Mars, Luna, an asteroid, or other objects in space. The architecture must be able to guarantee highly autonomous, reliable, and robust control over an extended time period with high level task-oriented teleoperation. The goal of achieving high autonomy within a remote supervisory control umbrella necessitates a distinctive architecture that integrates concepts originating in the intelligent control and communications networks areas. This presentation will discuss the architecture, the consideration that led to it, and the current state of its implementation.

MODEL-BASED HIGH AUTONOMY ARCHITECTURES

Autonomy is the ability to function as an independent unit or element over an extended period of time, performing a variety of actions necessary to achieve pre-designated objectives while responding to stimuli produced by integrally contained sensors. Emerging from the control field, intelligent control is viewed as a new paradigm for solving control problems [16]. However, its relatively narrow interpretation of the "control problem" does not fully accord with high autonomy requirements since it does not include the control needed by a system to diagnose and repair itself after significant insults to its physical structure. Requiring greater autonomy from a system forces the expanded view presented by Antsaklis and Passino [2] in their framework for autonomous control systems: full integration of knowledge-based reasoning (derived from artificial intelligence) together with perception and action components (derived from robotics and control).

To achieve such integration, Saridis [19] developed a three layer hierarchy (execution, coordination and organization) for intelligent control which is supposed to reflect increasing intelligence with decreasing precision. Antsaklis and Passino [2] refine the hierarchy to an arbitrary number of layers, depending on the particular application. The coupling of control and information at various layers characterizes the framework proposed by Albus [1]. At the core of Albus's "reference model architecture" is a world model, the intelligent system's "best estimate of objective reality." In such a model-based architecture, knowledge is encapsulated in the form of models that are employed at the various control layers to support the predefined system objectives. One major hurdle to be overcome in such architectural concepts is the heavy on-line computation times needed for higher level task-oriented functions such as planning and diagnosing. In high autonomy applications to planetary resource utilization a premium is placed on payload size and weight and thus (in the present state-of-the-art at least) computational load is a major concern. To overcome this obstacle, Zeigler and Chi [28] proposed a model-based architecture that seeks to reduce on-line computation by off-line pre-compilation to produce simplified models individually matched to the tasks needing to be performed. Such models are attached to generic engines that can interpret them efficiently with reduced processing times. As will be described later, the task-oriented models are developed as abstractions of a full-dynamics simulation model of the plant. Such models must be expressed in a variety of formalisms and at various levels of abstraction. Lower control layers are more likely to employ conventional differential equation descriptions and other forms of dynamic system models. Symbolic models derived from artificial intelligence (AI) research, such as logics for reasoning and planning, are more applicable at higher layers. A key

requirement for achieving high autonomy is the systematic development and integration of such dynamic and symbolic models. In this way, traditional control theory, where it is applicable, can be interfaced with AI techniques, where they are essential.

REMOTE SUPERVISORY CONTROL

Interaction with human operators at the very highest levels of a high-autonomy system provides the flexibility to deal with those rare but unavoidable events which are either impossible to foresee, or, if accounted for, would unacceptably increase the computational load and memory requirements [20]. To facilitate such remote supervisory control requires that a distributed command and control system enable operators to observe the ongoing in situ process and issue high level commands from one or several remote sites. Ultimately there will be many remote sites geographically distributed throughout the world. Such sites will communicate with each other and with the earthside command center over the national data communication network (currently the Internet). The command communication center will be located at the primary uplink, most likely at White Sands, New Mexico. This network will eventually also employ other uplinks, as well as various relay satellites orbiting the earth, Luna, and Mars. The space-based portions of this network will utilize optical links for increased bandwidth. At any time, there will be one operator in charge of each distinct plant, while other observers may watch from other remote sites. It should be possible to switch operations authority from one operator to another arbitrarily during an experiment without jeopardizing the overall mission. Moreover, if the remote controller or its connection to the plant fails then the latter should be able to continue on its own, possibly at a lower level of performance. When supervisory authority is reasserted by

another remote workstation the local plant controller should relinquish its full autonomy and resume operation under remote supervision.

Such a command and control system must be resilient to temporary breakdowns in communication links, and must be able to accommodate a varying number of remote participants and local plants to be controlled. New remote observers should be able to join at any time, while others may sign off. New plants should be attachable to the control umbrella at will. Conversely, those that have accomplished their missions or are no longer serviceable should be easily removable.

In what follows we shall first overview the command and control architecture that was developed to meet the foregoing requirements. Then we shall describe the model-based kernel that provides local high autonomous operation.

THE COMMAND AND CONTROL ARCHITECTURE

Figure 1 shows the overall command and control architecture. Each global site, whether local to the processing plant, or remote, hosts a Command Communication Center (CCC). On one hand, such a center serves as the gateway to the "longhaul" network that links this CCC to other CCCs; on the other hand, the center manages the resources available at the site. The reliability of the CCCs themselves can be guaranteed by standard technology such as resource duplication. Plants and operators communicate with their own CCC through an interface computer. For operators this interface computer is a VMS or Unix workstation running the Operations and Science Instrument System (OASIS) software developed at the University of Colorado [7,14]. OASIS has been successfully employed in a number of NASA applications.

OASIS is a layered software architecture developed specifically for remote supervisory control. It provides a convenient human-computer interface with color graphics, mouse or keyboard command entry, and multiple window displays of telemetry data from the plant site. OASIS itself controls data flows, translates mouse clicks and pull-down menu actions into command streams, receives and processes telemetry data, and controls the communication process using TCP/IP protocols. Lower levels (TAE+, Motif, X) provide window editing, management, and control. The software is database driven, so that programming new applications can be accomplished very quickly.

The operator workstations are called Remote Commanding Computer (RCC) and Remote Observing Computer (ROC) respectively. There is no fundamental difference between the two types of workstations. They run the same software. They differ only in the privileges given to the operator of the workstation at any time. The privileges are a resource that is maintained by the CCC. Additional privileges can be requested from the CCC at any point in time. The CCC will grant these privileges if the operator of the workstation has a sufficiently high priority, and if granting these privileges is not in conflict with other demands. For example, no plant should be commanded by several remote commanders simultaneously in an intrusive fashion. The reason for this is that, at least in initial operations capability, there will be no direct communication between the remote commanders. Thus allowing multiple simultaneous operation would lead to situations where remote commanders try to drive the plant toward inconsistent goals.

The operator privileges are symbolized in our current implementation by a privilege "key," that is maintained by the

CCC. This key can be requested by remote operators to establish them as the new remote commander. If the key has been requested, but is currently in use by another operator, the CCC will inform the current key holder of the request. It is then the responsibility of the key holder to relinquish the key when it is no longer needed.

The CCCs (local or remote) serve three purposes:

1. Software-decoupling the individual computers from each other. Each interface computer needs to know only the language of its client (the plant or the operator) and the language of the CCC. Different interface computers need not know anything of each other's characteristics and physical location, how many such computers are in the system, and how they operate.

2. Managing the resource umbrella of clients. Each CCC is responsible for managing the limited resources of its clients for example, a restricted resource of the RCCs is their privilege level, implemented through the privilege key in the current prototype. Restricted resources of the Local Controlling Computers (LCCs) may be the amount of energy to be used at any one time or the usable communication bandwidth between the LCC and its CCC.

3. Managing the communication with the other CCCs. Together, the CCCs manage the longhaul communication network. The time delays between the RCC/ROCs and their closest CCC will be short and a simple message acknowledging protocol can be employed. The duty of the CCCs will be to ensure proper transmission of commands and telemetry packets across the potentially less reliable longhaul network.

THE LOCAL INTELLIGENT CONTROL ARCHITECTURE

Having discussed the communications portion of the command and control architecture, we now come to the command structure. At the "local" site the plant communicates with its CCC through the Local Controlling Computer (LCC). As shown on Figure 2, the LCC comprises a distributed control architecture in itself [11]. The plant itself is interfaced with a hardware controller, called a Programmable Interface Unit (PIU) , that is responsible for implementing low-level control strategies. The PIU, a commercially available DataPac10K4T [8] has an advanced multiprocessor architecture and a fixed operating software that enables preprogramming to recognize and instantly respond to a number of simple commands. Several analog signal conditioner cards accommodate different types of transducers. The interface unit has internal registers for one thousand internal binary variables. These bits may be used for direct logic input/output (I/O) for process control, automatic triggering of executable commands, and initiation of limit-violation responses. It also has calibration functions for its I/O channels, limit status monitoring on user selectable channels, and digital/analog signal I/O functions. Numeric functions can be defined with one or more data channels as arguments. With a combination of the above functions, simple control programs can be downloaded to the interface unit from the LCC.

The LCC translates high-level task-oriented commands received from the RCC via the CCC into sequences of low-level commands, downloads the corresponding low-level control programs into the PIU, and initiates the control action by enabling the PIU control. Notice that, in Figure 1, the term LCC denotes the overall local control architecture, whereas in Figure 2, LCC is only a part of the local control architecture. This is

only for our convenience in showing details of the current implementation. In the generic representation (Figure. 1) the LCC could of course be implemented as one or several processors. The LCC is the heart of the autonomous control system at the site of the plant. The LCC communicates with its CCC to receive control parameters and to send telemetry data back to the RCC on Earth. Users' commands include commands to set control parameters, telemetry data requests, system start-up commands, and system shut-down commands. The LCC communicates with other computers/controllers while also executing the local control procedures. Thus the communication process is included as part of the control program. Figure 3 shows the overall protocol for the communication between the LCC and CCC. The LCC, physically a PC-386, first sets up communication links to the PIU in order to initialize the sensors and actuators of the plant. Then it tests the communication link to the CCC. Upon receipt of the control parameters and the start-up command from the CCC, the LCC begins to control the plant. Figure 4 shows the decision process for control of start-up and steady-state operations. In order to operate in real-time, the inference engine of the architecture has a clock to check time constraints on control rules. When the real-time expert system receives tasks from the RCC, it schedules the appropriate control actions to execute them. During the execution, it continually monitors the state of the plant.

The LCC and PIU together form a two level hierarchical control architecture. Reasoning and high-level logic are realized in the LCC in an expert system shell written in C. In contrast, the PIU contains simple control programs in memory and executes low-level control tasks in accordance with the parameters received from the LCC. This bi-level partition of functionality enables fast local control under the guidance of

slower, more global, intelligent control.

MARTIAN OXYGEN PRODUCTION PLANT

To provide a concrete example of the design methodology, we will describe the design of an oxygen production plant that would eventually operate on Mars. In keeping with the model-based architecture paradigm for high autonomy, we briefly describe the oxygen generation process and the full-dynamics model of the plant that we developed to support the design.

To exploit the Martian atmosphere, which is 90% carbon dioxide, a process of oxygen extraction has been studied by several investigators [13]. The process requires that input gas, at a pressure of 6 millibar and at a temperature of approximately 200K) be compressed (and thereby heated) to a pressure of 1 bar. It is then heated further to a temperature of approximately 800 K. At that temperature, carbon dioxide decomposes through thermal dissociation into carbon monoxide and oxygen. The heart of the oxygen production plant is an array of Zirconia tubes or disks which separate the two gasses in an electrocatalytic reaction. The oxygen is liquified and stored and the carbon monoxide is either discarded or converted to methane by a Sabatier process. Parenthetically, this process is also important for Lunar oxygen production, as many of the proposed processes (e.g., carbon ilmenite reduction), result in carbon dioxide as an intermediate product.

To develop a high fidelity simulation model of the oxygen production system we employed a modeling and design methodology based on Bond Graphs [4]. The Bond Graph formalism affords the ability to unify various processes in terms of energy flow relationships. For example, the power

flow in the thermal dissociation of carbon dioxide into oxygen and carbon monoxide can be formulated in a separate model which can be conveniently connected to the thermal model of the overall system. In this way we can construct a full-dynamic system model that can be used to represent both steady-state operation (flow equilibrium) as well as start-up and shut-down phases with high accuracy. Space limitations preclude a detailed presentation of the Bond Graph models. However, the general methodology is fully explained by Cellier [4] and application of the methodology to the oxygen plant is discussed in a technical report [10].

To meet the needs of the control and fault management tasks involved in autonomous operation, the full-dynamics simulation model is recast into various abstractions. Figure 6 illustrates three such abstractions, for use in tuning the parameters of a fuzzy-logic process controller, training a neural net diagnoser and testing the operation of a command interface, respectively. The abstractions are employed as fast running stand-ins for the base model in constructing the specific task engines [27,28]. To be useful surrogates, the abstractions should be valid simplifications of the full dynamics base model.

FAULT MANAGEMENT

Interspersed with the control tasks just described, the LCC reads sensory data from the PIU and reports telemetry data through the CCC to update the RCC's data base. If the LCC detects a faulty situation, it reports the fault detection to the RCC and starts its diagnosis inferencing engine. The behavior of each state variable is controlled by a dedicated watchdog monitor [5]. Each watchdog monitor has upper/lower limits for its state variable. Violation of these limits causes the watchdog monitor to activate diagnosis rules to

execute appropriate recovery actions. During the diagnosis process, the LCC sends data more frequently to the RCC, which represent the error state transition behavior of the plant. This alerts the user at the RCC to the deviant behavior of the plant. On such an occasion, or indeed, at any time, the RCC can route new commands or parameter values through the CCC to the LCC to re-schedule control tasks and/or to establish new control set points. Also, the RCC can send shut-down commands to interrupt the system. This can happen even while the system is in start-up mode. At the same time, if recovery is not possible, the expert system automatically executes a benign shut-down procedure to protect the plant from further damage.

The nature of extraterrestrial environments makes automated fault diagnosis an essential prerequisite for high-autonomy control of in situ plants. This fault diagnoser must be very reliable. Model-based diagnosers [13] have been demonstrated to provide high coverage of anticipated and unforeseen faults. However, since it is impossible to foresee all faults that might occur, it is desirable to build some redundancy into the fault monitoring and diagnosis procedures. For this reason, a second process fault diagnosis system using multiple sensors for data acquisition and neural networks for information processing has been developed [23].

The execution of this system is divided into three stages:

Item Fault Detection: At this stage a possible process fault is detected by checking if particular measurable or estimable variables are within a certain tolerance of the normal values. For example, the measured temperature of the Zirconia tube should be above 790K but below 815K under normal conditions. If this check is not passed, it

leads to a fault message that activates the next stage of fault diagnosis.

Fault Diagnosis: The fault is located and the cause of it is established at this stage using a neural network that fuses the data from several sensors. A multilayer feedforward net with one hidden layer was used. The reason to choose this type of networks is mainly due to its simplicity and available software. However, some recent studies have suggested that a multilayer feedforward network with a hyperbolic tangent as the nonlinear element seems best suited for the task of fault detection and diagnosis [22]. The input layer has ten nodes representing ten sensor readings, and the output layer has eight nodes - one for each of eight selected fault situations. The hidden layer has six nodes. The standard sigmoid was used as the activation function for the output neurons, while the hyperbolic tangent was used for hidden neurons in order to speed up the learning process.

Fault Evaluation: An assessment is made of how the fault identified in the second stage will affect the production process. The faults have been classified into different hazard classes according to a simple fault tree analysis. After the effect of the fault is determined, a decision on the actions to be taken will be made. If the fault is found to be tolerable, the production process may continue. If it is conditionally tolerable, a change of operation has to be performed by either modifying the local control algorithm or sending a request to the higher level of control for guidance. However, if the fault is intolerable, the process will be shut down immediately and an emergency request will be made to the

higher level to eliminate the fault. For example, if a malfunction in the heater has been determined to be the cause of high temperature, the operation will be stopped immediately and a request for changing the heater will be made.

To design the fault diagnoser ten sensor readings were used for sensor fusion in the neural network. One thermocouple transducer is located inside the Zirconia tube. On each of the CO2, O2, and CO2/CO pipes, one thermocouple, one pressure transducer, and one flow rate transducer are located. All readings are scaled to a range from -1.0 to +1.0. The scaling makes the sensor fusion easier to perform because the original measurement data contains both small and large values. Eight representative fault situations were chosen:
1) CO_2 valve partially opened; 2) O_2 valve partially opened; 3) CO_2/CO valve partially opened; 4) Thermocouple transducers broken; 5) Leak flow in tube; 6) Malfunction in heater;
7) CO_2 flow rate too high; and 8) CO_2 flow rate too low.

VERIFICATION AND VALIDATION

To assure that a complex high-autonomy system such as that described actually achieves its intended mission requires a comprehensive plan for verification and validation. Since our goal is to establish the proof-of-concept of the underlying command and control architecture, it is important that testing be sufficiently extensive to provide some confidence either that an actual system would perform as required. This section reports on the state of completion of the prototype, some of the tests of performance that were done, and some lessons learned from the experience.

A version of the prototype has been completed that demonstrates a respectable level of functionality for the communications and control portions separately. However, the overall system has not yet been put to test for reasons not related to the automation but to the development of the oxygen production plant itself. The highest priority for the latter is to demonstrate sufficient efficiency of production to suggest plausible space application. All efforts of the group in charge of its development are devoted to this end. At this time, we have had access to the plant on few occasions. On one such occasion, the operation of a rudimentary version of the architecture was demonstrated in a week long experiment with an actual prototype of the oxygen production plant. A truly unforeseen disturbance occurred during this period -- a thunderstorm caused a transient power outage. The ability of the controller to recover from this unplanned anomaly was notable and lends some credibility to the proposed fault management approach. The incident is described in [6].

Although the general outlines of the architecture have been verified, the specific incarnation as a command and controller of the oxygen production plant has yet to be fully tested. The reasons for this can be found in the delays encountered in our progress that caused departure from an ideal development of a model-based architecture. Such an ideal progression is predicated on the prior construction of the plant. Once the plant exists, a full dynamics base model is constructed and validated against the plant. Work on the abstractions intended to support various tasks can be started concurrently with development of the base model and validated against the latter when it has itself been validated against the real system. The task engines can be designed while the abstractions are validated and then tuned with the help of these abstractions after they have been validated. Once verified in this manner,

the engines can be tested against the plant individually and collectively within the completed command and control system.

SUMMARY AND FUTURE WORK

The basic outlines of the architecture for high autonomy command and control of space-based processing plants have been validated in the experimental work described. However, future work must extend and strengthen the model-based architecture methodology to apply to diverse processes and plant designs. Improved techniques and tools are required to facilitate development of faster running, more flexible models to support the design and tuning of task-oriented engines More advanced concepts in the higher levels of hierarchical planning, sensing, control, and exception handling must be integrated into the framework of the model-based architecture. Design for increased autonomy must emphasize graceful degradation of performance with reduced resource availability that arises when resources must be shared among commanders or as a consequence of system failure. This will require integration of computer vision and other advanced sensory capabilities (including sensor fusion) for world state assessment as well as fault detection, diagnosis, and recovery. For eventual deployment of in situ processing systems, the major challenge will be to reduce to practice the architectural concepts discussed here [5]. This will require addressing the tradeoffs between higher autonomy and remote supervision and between high component redundancy and intelligent self-diagnostic capability. These tradeoffs may be ameliorated by the increased computation and memory afforded by continuing development of light weight and low power components, but will continue to pose severe limitations in the foreseeable future.

ACKNOWLEDGMENT

This research was supported by the University of Arizona-NASA Space Engineering Center for Utilization of Local Planetary Resources. We wish to thank all of the many students whose efforts were essential to our progress but whose number is to large to list as co-authors of this article.

REFERENCES

1. J.S. Albus, "A Reference Model Architecture for Intelligent Systems Design", in: Antsaklis, P.J. and K.M. Passino (eds), An Introduction to Intelligent and Autonomous Control, Kluwer Academic Publishers, Norwell, MA. 1992.

2. P.J. Antsaklis, and K.M. Passino, "Introduction to Intelligent Control Systems with High Degrees of Autonomy", in: Antsaklis, P.J. and K.M. Passino (eds), An Introduction to Intelligent and Autonomous Control, Kluwer Academic Publishers, Norwell, MA. 1992.

3. K.J. Astrom, and K.E. Arzen, "Expert Control", in: Antsaklis, P.J. and K.M. Passino (eds), An Introduction to Intelligent and Autonomous Control, Kluwer Academic Publishers, Norwell, MA. 1992.

4. F.E. Cellier, Continuous System Modeling, Springer-Verlag, New York. 1991.

5. F.E. Cellier, L.C. Schooley, M.K. Sundareshan, and B.P. Zeigler, "Computer-Aided Design of Intelligent Controllers: Challenge of the Nineties," in: Recent Advances in Computer Aided Control Systems Engineering (M. Jamshidi and C.J. Herget, Eds.), Elsevier Science Publishers, Amsterdam. 1993.

6. F.E. Cellier., L.C. Schooley, B.P. Zeigler,A. Doser, G. Farrenkopf, J. Kim, Y. Pan, and B. Williams, "Watchdog Monitor Prevents Martian Oxygen Production Plant from Shutting Itself Down During Storm", in: Jamshidi M., (ed) "Robotics and Manufacturing: Recent Trends in Research, Education and Applications", Vol. 4, ASME Press, 1992 pp. 697-704.

7. R. Davis, and E. Hansen,"OASIS Teleoperations Package Makes its Debut," Information Systems Newsletter, 1986.

8. Daytronic Corp. System 10 DataPac Model 10K4T Instruction Manuals, Daytronic Corporation, Miamisburg, Ohio, 1990.

9. A. Doser, Multiple User Communications for Telescience, MS Thesis, Department of Electrical and Computer Engineering, University of Arizona, available also as: Technical Report TSL-028/92, Telescience Laboratory, Electrical and Computer Engineering Department, University of Arizona, Tucson, Arizona, 1992.

10. G. Farrenkopf, Full-Dynamics Simulator of the Martian Oxygen Production Prototype, Technical Report, SL-031/92, Telescience Laboratory, Electrical and Computer Engineering Department, University of Arizona, Tucson, Arizona, in preparation, 1992.

11. G. Farrenkopf and A. Doser, "Remote Command and Simulation of an Oxygen Production Plant," The SERC Newsletter, Volume 3, Number 1, pp. 5-6, 1992.

12. J.P. Hollman, Heat Transfer, McGraw-Hill, New York, 1996.

13. J.-K. Huang, M.-T. Ho, and R.L. Ash, "Expert Systems for Automated Maintenance of a Mars Oxygen Production System", J. Spacecraft and Rockets, Vol. 29., No. 4, pp. 425-431, 1992.

14. A. Jouchoux, E. Hillis, G. Tate, "Porting a Spacecraft Monitor and Control System Written in ADA," Proc. 6th Washington ADA Symposium, pp. 163-168, 1989.

15. C.C. Lee, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller - Part I," IEEE Trans. Sys. Man Cyber., Vol. 22, No. 2. 1990.

16. A. Meystel, "Intelligent Control: issues and Perspectives", Proc. IEEE Workshop on Intelligent Control, pp. 1-15, 1985.

17. OASIS System Managers Guide, University of Colorado, Operations and Systems Information Group, Laboratory for Atmospheric and Space Physics, Boulder, CO 1991.

18. R.H. Perry, and D.W. Green, Eds. Perry's Chemical Engineer's Handbook, McGraw Hill, 1991.

19. G.N. Saridis, "Analytic Formulation of the Principle of Increasing Precision with Decreasing Intelligence for Intelligent Machines", Automatica, Vol. 25, No. 3, pp. 461-467, 1989.

20. L.C. Schooley, and F.E. Cellier, "Monitoring and Control Systems for Automated Process Plants," Proceedings of the Invitational Symposium on Space Mining and Manufacturing,NASA Space Engineering Research Center for Utilization of Local Planetary Resources, Tucson AZ, 1989.

21. L.C. Schooley, F.E. Cellier, F.-Y. Wang, and B.P. Zeigler, "Intelligent Control and Communication Systems," Proceedings of the Invitational Symposium on Smaller, Cheaper, Faster Missions to the Moon and Mars, NASA Space Engineering Research Center for Utilization of Local Planetary Resources, Tucson AZ, 1993.

22. T.N. Sorsa, G. Koivo, and H. Koivisto, "Neural Networks in Process Fault Diagnosis," IEEE Trans. on Systems, Man, and Cybernetics, Vol.21, No.4, pp. 815-825, 1991.

23. F.-Y. Wang, "Building Knowledge Structure in Neural Nets Using Fuzzy Logic," submitted to Journal of Fuzzy and Intelligent Systems.

24. F.-Y. Wang and F. Wu, "Sensor Fusion and Process Fault Diagnosis for Martian Oxygen Production Plant Using Neural Nets," Technical Report 42, Robotics and Automation Laboratory, Department of Systems and Industrial Engineering, University of Arizona, 1993.

25. Q. Wang and F.E. Cellier, "Time Windows: Automated Abstraction of Continuous Models in Discrete-Event Models in High Autonomy Systems", Int. J. Gen. Sys., Vol. 19, No. 3, pp. 241-262, 1991.

26. B.P. Zeigler, "DEVS Representation of Dynamical Systems: Event-based Intelligent Control," Proceedings of IEEE, Vol. 77, No. 1, pp. 72-80, 1989.

27. B.P. Zeigler, Object-Oriented Simulation with Hierarchical, Modular Models, Academic Press, NY, 1990.

28. B.P. Zeigler, B.P., and S.D. Chi, "Model-Based Architecture Concepts for Autonomous Systems Design and Simulation", in: Antsaklis, P.J. and K.M. Passino (eds), An Introduction of Intelligent and Autonomous Control, Kluwer Academic Publishers, Norwell, MA, 1992.

29. B.P. Zeigler and J. Kim, "Extending the DEVS-Scheme Knowledge-based Simulation Environment for Real-time Event-based Control," to appear in IEEE Trans. on Robotics and Automation.