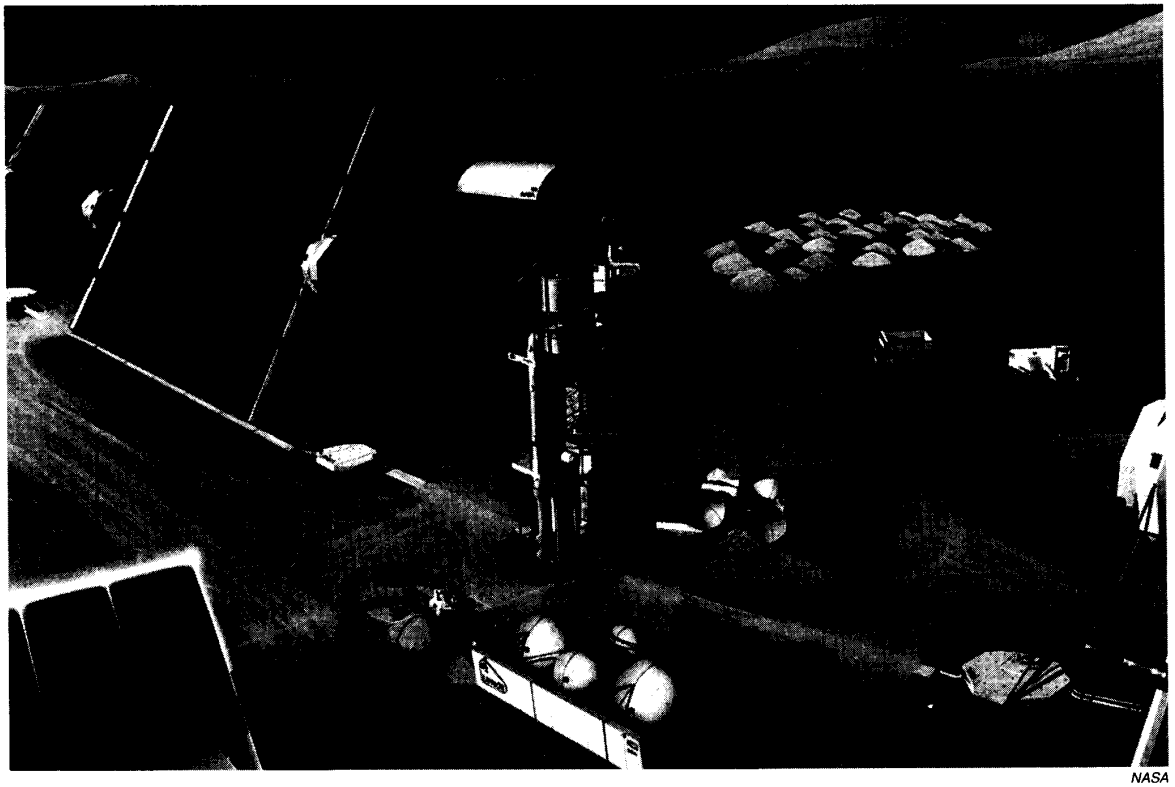
High-Autonomy Control of Space Resource Processing Plants

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



NASA

A high-autonomy intelligent command and control architecture has been developed for unmanned plants to conduct scientific experiments or process local planetary resources. Two applications are: a biotechnology laboratory designed for Space Station Freedom and a working prototype of a plant for producing oxygen from the Martian atmosphere. 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 testbed has been developed to demonstrate several applications of the architecture.

L.C. Schooley, B.P. Zeigler and F.E. Cellier are with the Department of Electrical and Computer Engineering, University of Arizona, Tucson, AZ 85721. F.-Y. Wang is with the Department of Systems and Industrial Engineering, University of Arizona, Tucson, AZ 85721. B.P. Zeigler's email address is zeigler@helios.ece.arizona.edu. *This work was supported by the University of Arizona-NASA Space Engineering Center for Utilization of Local Planetary Resources.*

High Autonomy within Remote Supervisory Control

A distributed command and control architecture has been developed to provide the capability to teleoperate one or several process plants located on Mars, Luna, the asteroids, and/or other objects in space in a supervisory control mode from one or several locations on planet Earth. The architecture is able to guarantee autonomous, reliable, and robust control over an extended time period with high-level task-oriented teleoperation.

 0272-1708/93/$03.00©1993IEEE

The goal of achieving high autonomy within a remote supervisory control umbrella necessitates a distinctive design that integrates concepts originating in the intelligent control and communication networks areas. Here we discuss the architecture, the considerations 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 predesignated 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 degrees of autonomy from a system forces the more 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 precompilation 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, NM. 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

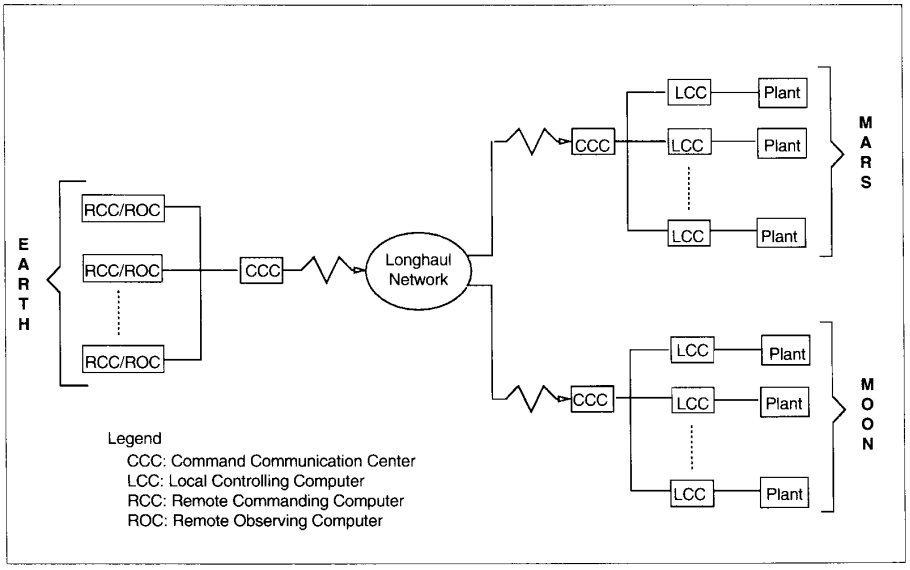


Fig. 1. Global remote command and control architecture.

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 highly autonomous operation.

Command and Control Architecture

Fig. 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 Computers (RCC) and Remote Observing Computers (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

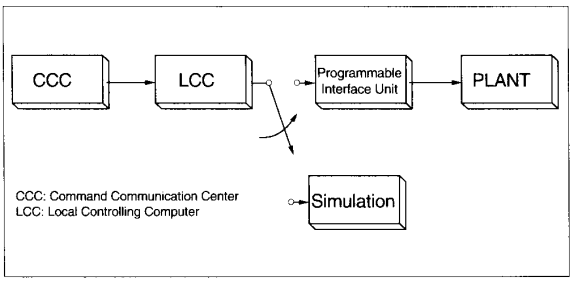


Fig. 2. Local control architecture.

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 of 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.

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 Fig. 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

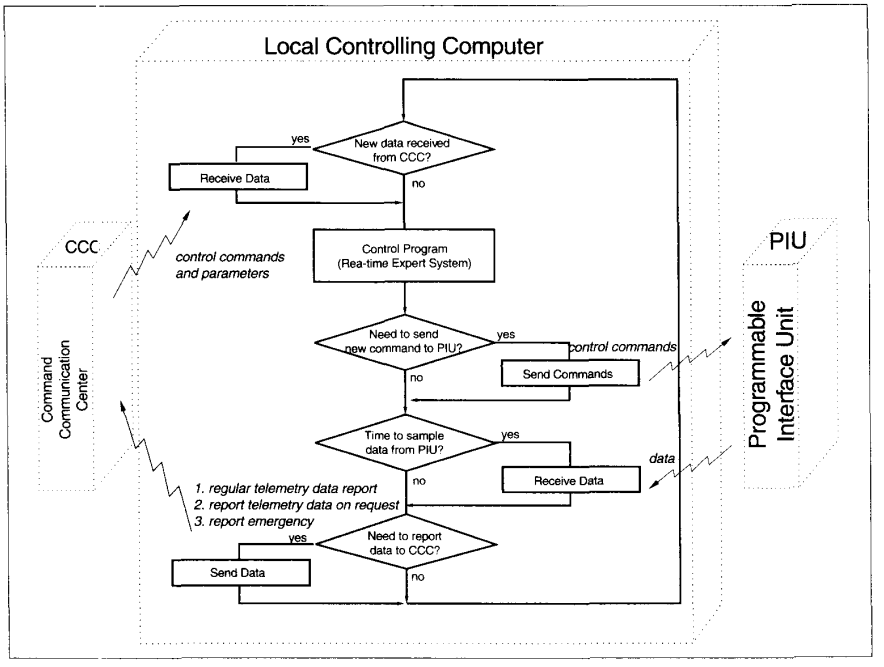


Fig. 3. Protocol in the LCC for communicating with the CCC and the PIU.

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 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 Fig. 1, the term LCC denotes the overall local control architecture, whereas in Fig. 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 (Fig. 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. Fig. 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. Fig. 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.

Full-Dynamics Model of a 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.

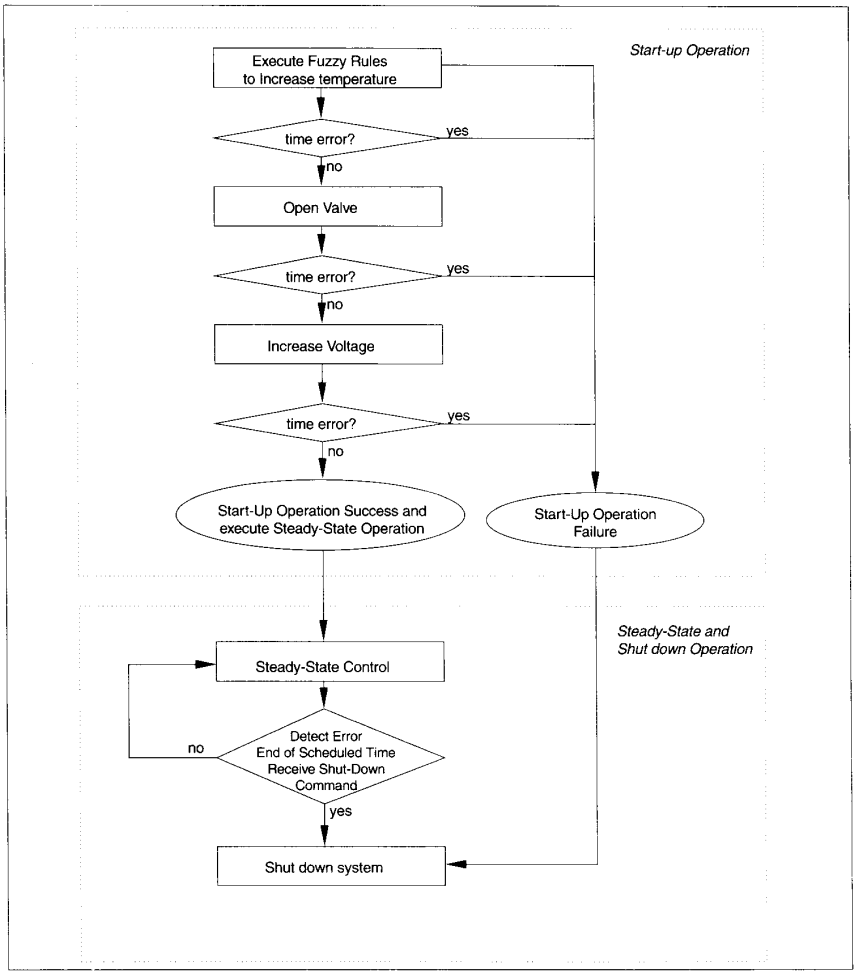


Fig. 4. Flow diagram of start-up and steady-state control scheme.

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]. The following gives a brief overview of the methodology and its application.

The oxygen production system consists of concentric cylinders of different materials that perform different tasks as shown in Fig. 5. A cylindrical ceramic heater surrounds a metal pipe and is surrounded by a cylinder of insulation. Carbon dioxide gas enters the system through an aluminum pipe and then passes over the surface of the surrounding Zirconia tube. To form the Bond Graph thermal model, the cylinders can be described in terms of thermal resistance and capacitance elements [12]. To conform to the Bond Graph requirement that all quantities be expressed in terms of power, the resistance to entropy flow is used instead of the resistance to heat flow. The temperature dependent relations for the specific heat of the different materials are available in the literature [18]. Some of the cylinders contain gas, which means heat flow due to convection as well as conduction must be accounted for. The radiation heat transfer from the pipe in contact with the heater to the Zirconia tubes is also modeled. Since the system is symmetric the temperature distribution along the cylinders is expected to be approximately uniform. Thus the cylinders were divided into at most two sections. This allows for a smaller, less complex model with better simulation response, yet with acceptable accuracy [10].

To model the thermal dissociation of carbon dioxide into carbon monoxide and oxygen, requires consideration of chemical, thermic, and hydraulic/pneumatic forms of power. The oxygen production rate is predicted using the number of moles of the reactant. This model is modularly connected to the tube model at the point where the input gas has been heated. The molar flow rate of oxygen produced is then multiplied by the efficiency of the Zirconia tube separation process, which is dependent on temperature and applied tube voltage.

Design of the Process Controller

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. Fig. 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.

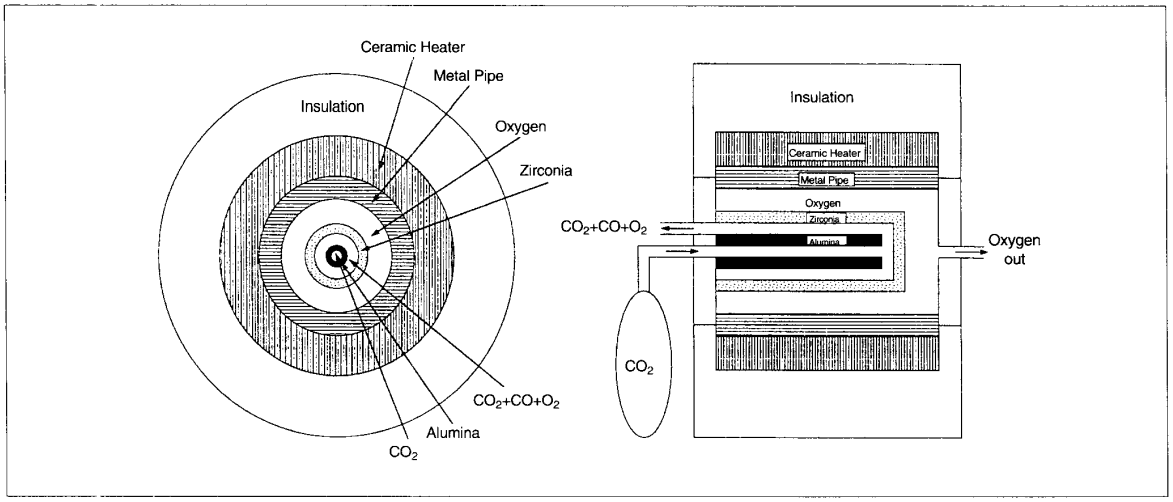


Fig. 5. Topology of zirconia tube.

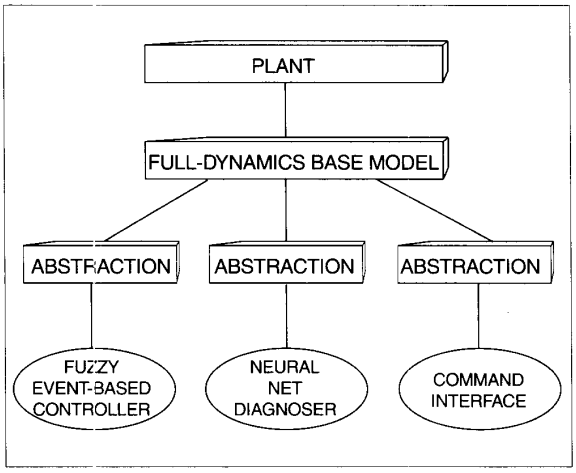


Fig. 6. Base model and abstractions for model-based architecture.

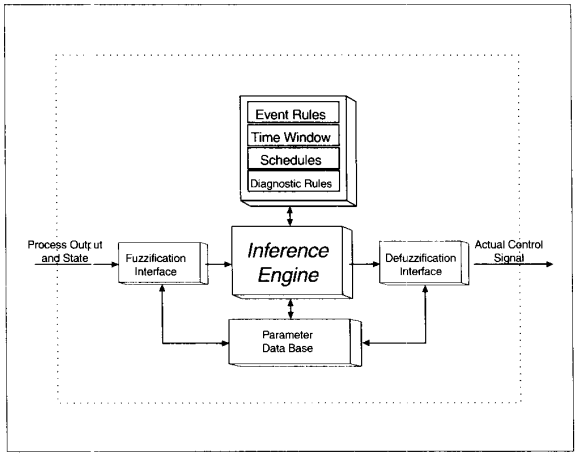


Fig. 7. Event-based fuzzy logic controller.

Event-Based/Fuzzy Logic Process Control

To illustrate this model-based approach, we first describe the design of the real-time process controller within the LCC. Three kinds of state variables of the plant must be controlled, viz., temperatures of the Zirconia tubes, flow rate of the CO_2 gas and voltages across the Zirconia tubes. Since the unit must operate at high temperature, a control algorithm is used to increase the temperature as fast as possible while still preventing the seal between metal pipe and Zirconia input tubes from breaking. To achieve such a temperature trajectory, a fixed rate of increase, e.g., 10°C/min, can be set as the goal rate for the control algorithm. It should be noted that the thermal system is highly non-linear and therefore not amenable to classical linear control theory.

With this in mind, an approach called event-based control [26] was integrated with fuzzy logic [15] to design the temperature controller. The structure of the controller is illustrated in Fig. 7. In event-based control, finite-state threshold sensors divide the control state space into a finite output partition. The control task is to move an initial state from a position on a given partition block boundary through a succession of boundaries to a goal set. Rather than continuously monitor the state trajectory, the controller has time windows in which it expects appropriate sensor responses to confirm expected boundary crossings. Such time windows can be developed as abstractions from a model of the controlled object [25]. A sensor response that occurs too early or too late indicates that an operations failure has occurred and causes the controller to cede control to a higher level process to deal with the abnormal situation. Moreover, the arrival of a sensor response outside its expected time-window provides important diagnostic information to a diagnoser tasked to discover the responsible fault, as the temperature example coming up will show. In sum, the event-based control paradigm provides a "seamless" transition from operations under normal conditions to fault-management after anomalies occur.

To implement event-based control, when the LCC receives a goal temperature from the RCC, the expert system sets a number of checkpoint temperatures between the initial temperature and goal temperature as shown in Fig. 8. A pair of successive check-

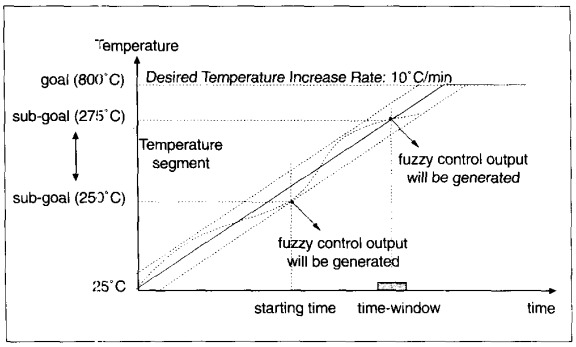


Fig. 8. Event-based temperature control.

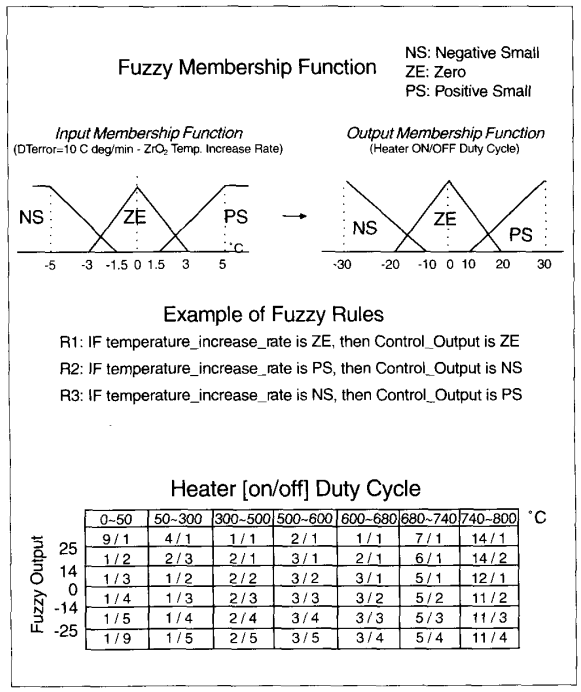
points can be considered as a small segment that must be successfully controlled to reach the final goal temperature. Event-based control is employed for each segment, i.e., after reaching one checkpoint and issuing a control command to reach the next, the latter should be reached within a predetermined time window. If the next checkpoint temperature is reached too early or too late, this represents a fault situation that the system needs to diagnose. To start its work, the diagnoser can use input from the event-based controller. For example, a late goal crossing suggests that the heater output is too low and starts a chain of reasoning in this direction. On the other hand, an early response might indicate a faulty thermocouple with a different sequence of considerations to initiate.

At each checkpoint, the fuzzy logic controller computes an appropriate control command, which is realized as a pair of on/off heater durations as shown at the bottom of Fig. 9 (for example, <2, 3> might mean heater on for 2 s, then off for 3 s.) High thermal mass in the plant necessitates this kind of operating regime (that is, the Zirconia temperature tends to lag and overshoot the nominal value). Fig. 9 also shows the fuzzy controller rules and the associated fuzzy membership functions for the rate of Zirconia temperature increase. The rate of increase is obtained by sampling the temperature at 1 s intervals. The breakpoints in these membership functions are parameters that can be adjusted to produce a temperature trajectory that closely matches that desired. Such adjustment can be supported by an appropriate abstraction of the base model to be discussed later.

The integration of event-based and fuzzy logic control paradigms provides a powerful means of meeting "hard" real-time subgoals, (offered by event-based control), while providing for flexible behavior-based "soft" control responses (offered by fuzzy control). Later in this article we discuss results of experimentation with this new approach and compare it with conventional alternatives.

Fault Management

Interspersed with the control tasks just described, the LCC reads sensory data from the PIU and reports telemetry data to 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



0~50	50~300	300~500	500~600	600~680	680~740	740~800
9/1	4/1	1/1	2/1	1/1	7/1	14/1
1/2	2/3	2/1	3/1	2/1	6/1	14/2
1/3	1/2	2/2	3/2	3/1	5/1	12/1
1/4	1/3	2/3	3/3	3/2	5/2	11/2
1/5	1/4	2/4	3/4	3/3	5/3	11/3
1/9	1/5	2/5	3/5	3/4	5/4	11/4

Fig. 9. Fuzzy membership functions and defuzzified control output.

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 shutdown commands to interrupt the system. This can happen even while the system is in startup mode. At the same time, if recovery is not possible, the expert system automatically executes a benign shutdown 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:

1. 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 situation. If this check is not passed, it leads to a fault message that activates the next stage of fault diagnosis.

2. 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 network is mainly due to its simplicity and available software. However, some recent studies have suggested that 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.

3. 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 emergent 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 CO_2, O_2, and CO_2/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.

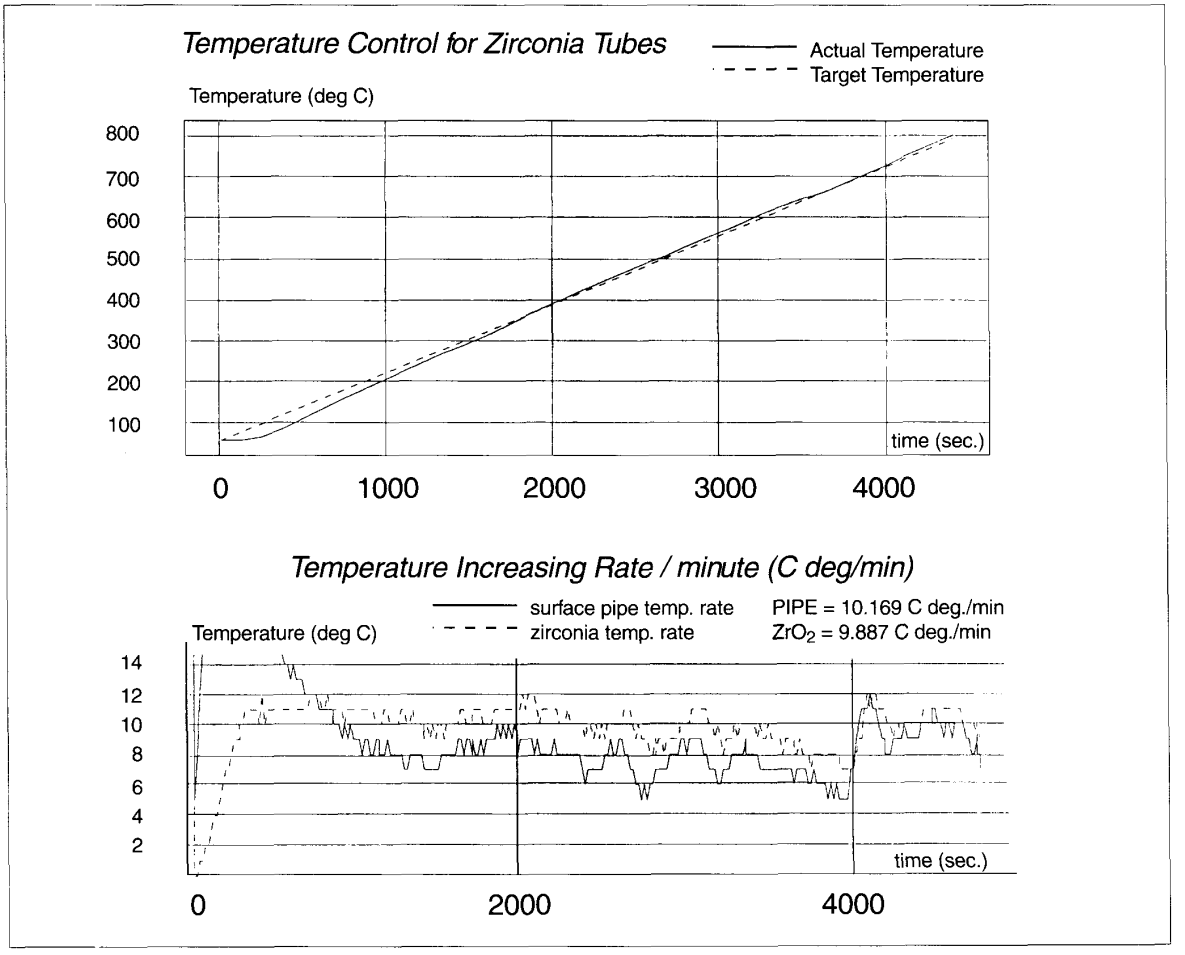


Fig. 10. Performance of event-based fuzzy logic controller.

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 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.

To date we have been able to follow this scenario mainly with respect to its development aspects. Thus, as indicated before, the full dynamics base model was completed as were several engines and abstractions. However, we have not been able to adhere very well to the validation aspects of the scenario. To clarify this distinction, we first describe progress in the development of task engines.

Event-Based/Fuzzy Logic Controller

The event-based/fuzzy logic temperature controller was tested on a plant prototype containing only one Zirconia tube. Fig. 10(a) shows that, after tuning on a discrete-event abstraction of the thermal system, the controller was able to raise the tube temperature so as to closely match the desired constant ramp increase of 10°C/min. Fig. 10(b) shows that, except at startup, the instantaneous error rate was confined within narrow bounds.

The performance of the event-based/fuzzy logic control compares well with alternative conventional techniques such as PID and PLC controllers [29]. One alternative that was investigated takes account of the extreme nonlinearity of the thermal system by partitioning the trajectory into three phases: initial startup, ramping up to the goal temperature, and regulation of the steady state. In each phase, a PID algorithm with different gains was employed. Such tuning of PID controllers is a major research direction in the area known as expert control [3]. In comparing the performance of the PID-based approach, we found that it was not able to match the accuracy in following the desired trajectory exhibited by the event-based/fuzzy logic controller. This is probably due to the large number of adjustable parameters and the inherent nonlinearity of the fuzzy logic scheme.

Although a PID-based algorithm can be designed to implement a given control strategy, such as the preceding temperature example, its structure is fixed and only its parameters can be modified to meet a non-linear or changing environment. Adaptive controllers add a second level of control that can adjust such parameters, but work within the imposed structure. By contrast, event-based/fuzzy logic control, is extremely flexible in that its basic structure can be readily adjusted by reprogramming. Moreover, discrete-event control is needed to command certain kinds of discrete actions such as start-up, shutdown, turning on/off valves, etc. This kind of control is beyond the reasonable capabilities of continuous state controllers.

Neural Net Diagnoser

The neural net diagnoser was trained on an abstraction of the full-dynamics base model that included both normal and abnormal behavior. The training data obtained from the model consisted of 450 measurement patterns (each pattern contains 10 sensor readings), for the normal operation and for the fault situations such as a valve partially opened, sensor or device malfunction, or abnormal gas flow rate. The output nodes each represented a fault. For the normal operation, the network was required to produce a value near zero (<0.1) at all the output nodes. whereas the presence of a fault should be indicated by a value near 1 (>0.9) at the corresponding output and near zero (<0.1) at the rest of the output nodes. The learning was conducted using the conjugate back-propagation algorithm [22]. The network was trained 10 000 times to learn the specified fault situations. For 100 additional measurement patterns that are not in the training data, 83 patterns were classified correctly [24].

Communications Prototype

The operation of the communications prototype was verified using a mock geographic distribution of workstations playing the roles the various computers. A simplified abstraction of the plant oriented to testing the command interface was employed. This abstraction, the LCC, and the local CCC were located in a mechanical engineering laboratory across the University of Arizona campus from a computer engineering laboratory where the RCC and several ROCs were located. The ME and CE labs thus played the roles of a planet (local site) and Earth (remote site) respectively. Communication was via campus ethernet using TCP/IP protocols. Since the remote CCC was not implemented, the local CCC managed the resources of the plant and also those of the remote operators [9]. Actual remote observation and control of the simulated plant was successfully demonstrated.

Model Validation

Now we return to our difficulties in the validation of the model-based architecture. Based on the Bond Graph methodology and

integrating complex thermal, chemical and electrical processes, a full-dynamics simulation model has been completed. This base model runs on a 386 PC approximately an order of magnitude slower than real time. It has not yet been fully validated against the real plant, nor have the task-oriented models abstractions been validated against it. Moreover, as already indicated, the oxygen production plant has evolved through several prototypes, and is still under development. As a consequence of such prevarication in the underlying plant design, developing a base model, and therefore all its subsequent derivatives, is not only subject to delay but is also an exercise in hitting a moving target.

Although the methodology is not proceeding in the ideal manner, it is still possible to claim benefits for it. Without the approximate realism afforded by the base model and its abstractions, design of the plant control system could not proceed until the plant had actually been constructed. The existence of these models facilitates early design of, and experimentation with, the task engines in a realistic test bed environment without incurring the cost and risking the destruction of actual equipment (this early start capability has become known as concurrent engineering in manufacturing literature). Therefore our response to "life in the real world" is to try to build in greater flexibility in the design of the base model, its abstractions, and the task engine structures so that when finally a firm plant comes into existence, the models and engines have a greater chance of being calibrated to it. For example, rather than optimize the parameters of the fuzzy controller against a single model, it can be optimized against a family of models representing the uncertainty ranges in plant parameter values.

Future Directions

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 faster 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 light weight and low power but will continue to pose severe limitations in the foreseeable future.

Acknowledgment

The authors wish to thank the many students whose efforts were essential to our progress but who are too numerous to list as co-authors of this article. Jinwoo Kim's help in preparing this article deserves special mention.

References

[1] J.S. Albus, "A reference model architecture for intelligent systems design," in *An Introduction to Intelligent and Autonomous Control*, P.J. Antsaklis and K.M. Passino, Eds. Norwell, MA: Kluwer, 1992.

[2] P.J. Antsaklis and K.M. Passino, "Introduction to intelligent control systems with high degrees of autonomy," in *An Introduction to Intelligent and Autonomous Control*, P.J. Antsaklis and K.M. Passino, Eds. Norwell, MA: Kluwer, 1992.

[3] K.J. Astrom and K.E. Arzen, "Expert control," in *An Introduction to Intelligent and Autonomous Control*, P.J. Antsaklis and K.M. Passino, Eds. Norwell, MA: Kluwer, 1992.

[4] F.E. Cellier, *Continuous System Modeling*. New York: Springer-Verlag, 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. Amsterdam, The Netherlands: Elsevier, 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 *Robotics and Manufacturing: Recent Trends in Research, Education and Applications*," M. Jamshidi, Ed., vol. 4. ASME Press, 1992, pp. 697-704.

[7] R. Davis and E. Hansen, "OASIS teleoperations package makes its debut," *Info. Syst. Newsl.*, 1986.

[8] Daytronic Corp. System 10 DataPac Model 10K4T Instruction Manuals, Daytronic Corporation, Miamisburg, OH, 1990.

[9] A. Doser, Multiple User Communications for Telescience, M.S. thesis, Dept. of Elec. and Comp. Eng., Univ. of Arizona; also available as Tech. Rep. TSL-028/92, Telescience Laboratory, Elec. and Computer Eng. Dept., Univ. of Arizona, Tucson, AZ, 1992.

[10] G. Farrenkopf, "Full-dynamics simulator of the martian oxygen production prototype," Tech. Rep. SL-031/92, Telescience Laboratory, Elec. and Computer Eng. Dept., Univ. of Arizona, Tucson, AZ, 1992.

[11] G. Farrenkopf and A. Doser, "Remote command and simulation of an oxygen production plant," *SERC Newsl.*, vol. 3, no. 1, pp. 5–6, 1992.

[12] J.P. Hollman, *Heat Transfer*. New York: McGraw-Hill, 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 Rockets*, vol. 29., No. 4, pp. 425–431, 1992.

[14] A. Jouchoux, E. Hillis, and G. Tate, "Porting a spacecraft monitor and control system written in ADA," in *Proc. 6th Washington ADA Symp.*, 1989, pp. 163–168.

[15] C.C. Lee, "Fuzzy logic in control systems: Fuzzy logic controller — Part I," *IEEE Trans. Syst. Man Cyber.*, vol. 22, no. 2, 1990.

[16] A. Meystel, "Intelligent control: Issues and perspectives," *Proc. IEEE Workshop on Intelligent Control*, 1985, pp. 1–15.

[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*. New York: 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," in *Proc. NASA Invitational Symp. Space Mining and Manufacturing*, Tucson, AZ, 1989.

[21] L.C. Schooley, F.E. Cellier, F-Y Wang, and B.P. Zeigler, "Intelligent control and communication systems," in *Proc. NASA Inv. Symp. Smaller, Cheaper, Faster Missions to the Moon and Mars*, Tucson, AZ, 1993.

[22] T.N Sorsa, K. Koivo, and H. Koivisto, "Neural networks in process fault diagnosis," *IEEE Trans. Syst., Man, and Cybern.*, vol. 21, No.4, pp. 815–825, 1991.

[23] F.-Y Wang, "Building knowledge structure in neural nets using fuzzy logic," submitted for publication.

[24] F.-Y Wang and F. Wu, "Sensor fusion and process fault diagnosis for martian oxygen production plant using neural nets," Tech. Rep. 42, Robotics and Automation Laboratory, Department of Systems and Industrial Engineering, Univ. 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. Syst.*, vol. 19, no. 3, pp. 241–262, 1991.

[26] B.P. Zeigler, "DEVS representation of dynamical systems: Event-based intelligent control," *Proc. IEEE*, vol. 77, no. 1, pp. 72–80, 1989.

[27] B.P. Zeigler, *Object-Oriented Simulation with Hierarchical, Modular Models*. New York: Academic, 1990.

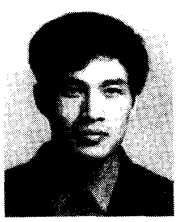
[28] B.P. Zeigler and S.D. Chi, "Model-based architecture concepts for autonomous systems design and simulation," in *An Introduction of Intelligent and Autonomous Control*, P.J. Antsaklis and K.M. Passino, Eds. Norwell, MA: Kluwer, 1992.

[29] B.P. Zeigler and J. Kim, "Extending the DEVS-scheme knowledge-based simulation environment for real-time event-based control," *IEEE Trans. Robot. Auto.*, 1993.

Larry C. Schooley was born in Wichita, KS, on September 21, 1938. He received the B.S. degree in electrical engineering with highest distinction from the University of Kansas, Lawrence, in 1961; the M.S. degree from the University of Maryland, College Park, in 1965; and the Ph.D. degree, with honors, from the University of Kansas in 1968. From 1961 to 1965 he was with the Naval Reactors Division of the Atomic Energy Commission where he supervised development of instrumentation and control equipment for nuclear propulsion of submarines and surface vessels. From 1965 to 1968 he was an NSF Engineering Trainee and Instructor of Electrical Engineering at the University of Kansas. Since then he has been a member of the faculty of the Electrical and Computer Engineering Department at the University of Arizona. He is a registered professional engineer, and a member of Eta Kappa Nu, Sigma Tau, Tau Beta Pi, the IEEE, and the American Society for Engineering Education. His research interests are in digital communications systems and networks, and in remote supervisory control of intelligent systems.

Bernard P. Zeigler is Professor of Electrical and Computer Engineering at the University of Arizona, Tucson. He received the B. Eng. Phys. degree from McGill University in 1962, the M.S.E.E. degree from M.I.T. in 1964, and the Ph.D. degree from the University of Michigan in 1969. He has published over 100 journal and conference articles in modeling and simulation, knowledge based systems, and high autonomy systems. His first book *Theory of Modelling and Simulation* (Wiley, 1976) is regarded as one of the foundational works in the field. A second book *Mutifacetted Modelling and Discrete Event Simulation* (Academic Press, 1984), was given the Outstanding Simulation Publication Award by TIMS College on Simulation in 1988. His current research on simulation methodology is described in a new book *Object-Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems* published by Academic Press, Boston, 1990. His research has been supported by federal agencies including NSF, NASA, USAF, and the U.S. Army, as well as industrial sponsors including Siemens, McDonnell Douglas, and Motorola. He is founder of a university spin-off company for technology transfer of his simulation environment concepts and the recipient of an Small Business Innovative Research grant from the U.S. Army.

François E. Cellier received the B.S. degree in electrical engineering from the Swiss Federal Institute of Technology (ETH) Zürich in 1972, the M.S. degree in automatic control in 1973, and the Ph.D. degree in technical sciences in 1979, all from the same university. Following the Ph.D., he worked as a Lecturer at ETH Zürich. He joined the University of Arizona in 1984 as an Associate Professor. His main scientific interests concern modeling and simulation methodology, and the design of advanced software systems for simulation, computer-aided modeling, and computer- aided design. He designed and implemented the GASP-V simulation package, and he was the designer of the COSY simulation language, a modified version of which, under the name SYSMOD, has become a standard at the British Ministry of Defence. He has authored or co-authored more than sixty technical publications, and he recently published his first textbook on *Continuous System Modeling* (Springer-Verlag, 1991). He served as chair of the National Organizing Committee of the Simulation'75 conference, and as chair of the International Program Committee of the Simulation'77 and Simulation'80 conferences. He recently served as the program chair of the 1993 International Conference on Bond Graph Modeling, and he will serve as the general chair of the 1994 Computer-Aided Control Systems Design conference (IEEE/IFAC). He is associate editor of several simulation related journals, and he served as vice-chair of two committees on standardization of simulation and modeling software.



Fei-Yue Wang was born in Qingdao, China, on November 2, 1961. He received the B.E. degree in chemical engineering from Shandong Institute of Chemical Engineering from Shandong Institute of Chemical Engineering, Qingdao, China, the M.S. degree in mechanics from Zhejiang University, Hangzhou, China, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, in 1981, 1984, and 1990, respectively. From 1984 to 1986 he was an Instructor at the Department of Mechanics, Zhejiang University. In 1986 he was awarded the Pao Yu-Kong and Pao Zao-Long Scholarship for Chinese Students for his academic achievements. From 1988 to 1990 he was with the NASA Center for Intelligent Robotic Systems for Space Exploration at Rensselaer Polytechnic Institute. He jointed the University of Arizona, Tucson, AZ, in 1990, where he is presently an Assistant Professor of Systems and Industrial Engineering. In his previous work in mechanics and applied mathematics he contributed significantly to the theoretical development of shells, plates, planes, three-dimensional elasticity, and micropolar elasticity for both isotropic and anisotropic materials. He is the co-author of the forthcoming book *Coordination Theory of Intelligent Machines: Applications in Intelligent Robotic Systems and CIM Systems* and has published more than 40 refereed journal articles and book chapters. He is the co-editor of the special issue on fuzzy logic and neural networks for the *Journal of Intelligent and Fuzzy Systems* and the founder of Synergetic Systems, Inc. His fields of interest include mechatronics, robotics and automation, computer vision, computer-integrated manufacturing, intelligent controls, and intelligent systems. He is a member of Sigma Xi, Chinese Society of Mathematics and Mechanics, the Association for Computing Machinery (ACM), and the American Society of Mechanical Engineers (ASME).