Enhancing Discrete Event Simulation by Integrating Continuous Models

Jane T. Malin

Automation, Robotics and Simulation Division
NASA Johnson Space Center
Houston, TX 77258
malin@jsc.nasa.gov

Land Fleming

Hernandez Engineering
Houston, TX 7708
fleming@mickey.jsc.nasa.gov

Abstract

This paper describes CONFIG, an implemented hybrid modeling and simulation approach, within a discrete event system framework, and its application to control software validation. CONFIG system models are made up of connected devices and activities, and device models are made up of mode transition models, with behavior models for each mode. Behavior models can be discrete or continuous, and characterize how changes in discrete inputs produce changes in outputs and mode transitions. In CONFIG, two operators, Integrate and Apply-When, are used to compute states or time advances that depend on continuous changes. The Apply-When operator calls external algebraic functions to determine the time advance for a rate-dependent event. The Integrate operator uses a discrete-time approach, providing periodic updates of variables. CONFIG simulations were used to validate advanced control software for gas transfer between chambers during a 90-day manned test of technology for Lunar-Mars life support.

Introduction

Operation of production plants for life support or propellant production in space requires a combination of discrete and continuous control. Physico-chemical or biological processors concentrate and convert gases and liquids. While these dynamic processes are active, they may require continuous regulation. When duty cycles require reconfiguration of these processors and of the system of storage and transportation equipment between processors, discrete control is typically required. This combination of discrete and continuous control presents a challenge to modelers and designers of controllers, because design solutions draw from two paradigms. In the dominant hybrid systems paradigm, design is based on continuous systems, with discrete extensions. In the other hybrid systems paradigm that arises from discrete manufacturing and reactive systems, design is based on discrete systems, with continuous extensions. A powerful control system would integrate both paradigms.

Our work on the CONFIG discrete event modeling and simulation tool represents the discrete paradigm. A substantial portion of intelligent autonomy software performs discrete reactive control. The focus is on executing procedures to configure and reconfigure systems, and on detecting faults and failures during these operations. For control, system modes are defined to describe the major operational configurations of the system (e.g., shutdown, standby, processing). Procedures change operating modes of system components (devices) to support the system mode being established. Otherwise, control is typically event-driven. Use of continuous control is relatively infrequent. Because the control focus is on establishing system modes, analysis often focuses on system-wide effects of events. When an event occurs because of a control action or a failure, effects can cascade through the system configuration. A valve that fails to open when commanded will have local and global effects on the configuration and the processors. Discrete event simulation is well suited for this type of analysis.

In discrete event simulation, behavior of devices is typically abstracted to correspond to functional operating modes or states. Transitions among device operating modes are events that are typically caused or prevented by state changing events, including external commands and externally or internally caused failures. Behavior within such modes is typically represented as constant, with simple pass-throughs of variable values. Within some operating modes, some devices exhibit constant rate behavior. A minority have variable rate behavior. In

discrete event simulation, the rate-related variable is typically the time advance from one event to another. Rather than variable changes being time-stepped (discrete time), time advances are event-stepped (discrete events).

The discrete event system specification (DEVS) formalism introduced by Zeigler (1976) specifies a system, with a continuous time base but discrete inputs and state transitions, as a structure:

$M = [X, S, Y, d_{int}, d_{ext}, \lambda, ta]$ where

X is the input value set (value-change events arbitrarily separated from each other in time);
S is the sequential state set;
Y is the output value set (value-change events);
d_{int}: $S \rightarrow S$, the internal transition function;
d_{ext}: $Q \times X \rightarrow S$, the external transition function,
where Q is the total state set = $\{(s, e) / s \in S, 0 \leq e \leq ta(s)\}$;
ta: $S \rightarrow R^{+}_{0,\infty}$, the time advance function (a positive real number for time to the next internal transition, computed upon transition into the state);
λ: $S \rightarrow Y$, the output function (value-change produced just before transitioning to the next state).

The time advance can be a function of d_{int}, d_{ext}, or λ. Asynchronous changes in these systems are managed by an event scheduler with a variable time advance.

Discrete event models are hierarchical models, composed of coupled interconnecting components. The connections are defined by the couplings of inputs and outputs between the components. The behavior of the coupled model is a result of the coupled behavior of its components. In the CONFIG extension of DEVS, the model structure can be "recomposed" during a simulation as the direction and activation of interconnections changes, changing the couplings.

In CONFIG the state space is further structured using modes. Within modes, according to a behavior model, state changes due to rate-influenced processes can occur at regular intervals in time or value, or state changes can be triggered asynchronously by an external change. Transitions between modes are based on a conditions or boundaries that are associated with a change in the model of the behavior of the device. In DEVS, boundary-based multimodels handle sets of complementary models that describe processes that have phases of qualitatively different behavior. Switching between models is handled by the finite state machine that manages phase transitions at boundaries. In CONFIG these phases transitions can occur within or between modes of devices.
Continuous models are sometimes needed to handle continuous behavior within operating modes that leads to events. The problem is to embed hybrid system modeling in a simulation system that is designed to operate in a discrete event framework, where time advances in varying step sizes from scheduled event to scheduled event (Zeigler and Praehofer, 1998). Within a small subset of the operating modes, a different continuous model may be applicable to each of a set of qualitative states or phases of behavior. Events in hybrid systems are not just scheduled in time, but are rather conditions reached as results of changing states, or "state events".

It is possible to incorporate a discrete-time approach, providing periodic updates of variables within an operating mode, but such an approach can trade performance against "missing" a critical state event. In a more natural approach, continuous models can be used to calculate the time advances to critical state events in continuous systems. Mechanisms can be developed to handle external events that interrupt or change rates and require recalculation of a time advance. Such capabilities have been developed in the CONFIG extension of discrete event simulation. Another approach to integrating continuous and discrete event simulation that is consistent with the event-oriented basis of discrete event simulation, is to quantize variables and calculate the variable time advances associated with quantized changes (Zeigler, 1998). There should be benefits from combining all these approaches.

CONFIG Extension of Discrete Event Simulation

CONFIG was developed to support analysis of designs for systems and their operations (Malin, Basham, and Harris 1990; Malin and Leifker 1991; Malin, Ryan, and Fleming 1993, 1994). CONFIG extends discrete event simulation with capabilities for continuous system modeling. The purpose of these enhancements has been to make it possible to apply discrete event technology for model-based prediction, to support design and evaluation of intelligent software for control and fault management. Although discrete event simulation has typically been used for stochastic analyses of scenarios, CONFIG simulations are deterministic, for specific states and inputs.

CONFIG uses a state transition system formalism in a system model made up of a set of connected components, or "devices" structured within a configuration or "flow path". The direction of physical flows and the effects of

flow reconfigurations are efficiently analyzed during simulations. Two of the basic building blocks of a CONFIG model are devices and activities. Devices model the behavior of system hardware components and activities model actions in procedures or software. Examples of system devices are pumps, valves, tanks and condensers. Device relations represent the connections between system components. Activity-device relations are used to relate activities to system components for control and monitoring purposes.

The modular discrete event modeling approach provides a framework for organizing and managing the application of more detailed knowledge. In device models, time-related behavior models are embedded within modes, and these modes are within state-transition systems. Two modes of a simple valve, for example, might be open and closed. The way a device interacts with connected devices can depend on the current mode. Failures can be modeled as modes or as factors that precipitate or prevent transitions. Transitions between device modes can be determined by control variables, variable changes propagated through inter-device connections, or by changes in system flows. Activity models are also state-transition models. Several levels of control can be modeled as activities. An activity might be used to control the positioning of a set of valves, for example. States of activity models, called activity phases, have embedded control behaviors. These behaviors can represent discrete or continuous control regimes, or elements of schedules or simulation scenarios.

Life support system applications have required accurate accounting of resource inventories transferred by continuous flow at variable rates to various locations within the modeled system. In CONFIG, two operators, Integrate and Apply-When, are used to periodically compute states or time advances that depend on continuous changes. The Apply-When operator calls external algebraic functions to determine the time advance for a rate-dependent event. The Integrate operator uses a discrete-time approach, providing periodic updates of variables.

The Apply-When Operator

The primary value returned by the Apply-When operator is the time of an anticipated event. The operator is supplied with the definitions of two functions: a normal-call function and an interruption-call function. The operator invokes the specified normal-call function to compute the values of a set of state variables and the time at which the values are to be assigned given the arguments passed by the operator to the function. The operator then schedules the assignments of the computed variable values at the time determined. If any of the state variables passed as arguments to the normal-call function change value prior to the time of the scheduled value assignments, the computed values will in general no longer be valid. The interruption-call function is then invoked to compute the values of the assignment variables at the time of the interruption. In addition to all the arguments passed to the normal-call function, the interruption-call function must accept as an argument the increment of time that has passed since the invocation of the normal-call function. The Apply-When operator then removes the invalidated assignment event from the simulator's schedule and the normal-call function is re-invoked to compute a new set of variable values and a new event time.

The Integrate Operator

The Apply-When operator is best suited for representing continuous processes internal to a device that are largely unaffected by interactions with other devices and for which the intermediate values of the continuous variables are not of interest. The Integrate operator was devised for representing continuous processes in a device that are extensively affected by interactions with other devices and for which a simulation of continuous change in variable values is of interest. While the Apply-When operator may be used to describe arbitrarily complex behavior, this is determined by the functions it invokes rather than the operator itself. In contrast, the Integrate operator permits complex behavior to emerge from the interaction of devices having relatively simple descriptions of internal continuous processes. The Integrate operator, when included in the description of a device's internal behavior, performs a simple linear approximation of the continuously changing quantity to be integrated.

For example, the pressure differential between two tanks causes gas to flow from the higher-pressure tank to the lower-pressure tank if a valve connecting them is opened. The descriptions of the mass of gases in both tanks may be represented as:

mass <- INTEGRATE *(mass mass.rate update.interval)*

where *mass* is the integrating variable and the *update.interval* is the time interval over which the rate is assumed to be constant.

Interactions of the two devices may cause the rate of flow to vary in complex ways that are not explicitly represented in the internal behavior descriptions of any of the devices. The *mass.rate* out of the high-pressure tank is always the same as the *mass.rate* into the low-pressure tank, but this

rate varies continuously as the pressure differential between the two tanks decreases. CONFIG's data-driven simulation engine causes a new pressure differential to be computed after each update of the gas masses, resulting in a smaller *mass.rate* and therefore a smaller increment of mass transfer on the next update of the two tanks.

The exponential decay in mass flow rate is easily approximated in this manner, even though the exponential relationship is not stated in a device description. The approximation to exponential decay is, of course, more accurate for smaller update intervals. However, regardless of the accuracy of the flow rate computed, the mass balance is at all times maintained to the precision of the floating-point facilities of the programming environment. In the enclosed life-support system simulations, accurate mass balances are of much greater importance than maintaining accuracy in the rates of flow. Mass cannot appear or disappear from the system as a simulation artifact. For simulated operations spanning many days, considerable accuracy can be sacrificed in the representation of flow rates as a function of time. By choosing large update intervals, simulation speed can be increased with no ill effects on the usefulness of the simulation.

Several enhancements have been implemented to make the Integrate operator flexible and useful.

Simulation Performance and Constraints on Time Interval Selection. It was frequently desirable to specify different update intervals for interacting devices. In the example of the two connected tanks, if one is much larger than the other, the pressure in the larger tank may vary almost imperceptibly over the same time interval in which the pressure in the smaller tank varies significantly. Constraining the update interval for the larger tank to be the small value chosen for the smaller tank would force recomputations of pressure of unneeded accuracy for the larger tank, serving only to reduce simulation performance. However, choosing different intervals would cause flow rates to differ between the two devices, and would therefore violate the mass balance. This problem was addressed by adding a mechanism to the Integrate operator that accounts for changes of rate between scheduled updates of the integrated variable. When the variable is finally updated at the scheduled time, the new value is based on all rates of flow in effect since the last update.

Changing Update Intervals. Conditions may be specified to change the Integrate update interval at any time during simulation, to greater or smaller values. This is useful when greater accuracy is needed while software interacts with the simulation at a control point, or when a threshold value of the integrated variable is approached. It is also useful when the magnitude of a rate changes substantially.

Limiting Values and Time Interval Selection. It can sometimes be difficult to choose a fixed update interval that will not violate a physical constraint of the real system. For example, in a system consisting of a tank from which gas is being vented to space, use of an update interval that is too large will result in the tank taking on a negative mass. Avoiding such overshoots can be troublesome to the modeler. If a sufficiently small update interval is chosen, the overshoot will not occur. But determining what the interval should be can be time-consuming. The interval selected may result in unneeded accuracy and degradation of performance during most of the simulation.

A modified form of the Integrate operator, Integrate-Within-Limits, allows specification of limits that are not to be violated. If scheduling an update of the integrating variable at the specified time interval would cause the variable to exceed the specified maximum or fall below the specified minimum, the operator instead schedules the next update for the time at which the variable would attain the limiting value at the current rate of change. Changes of rate are accounted for in the same way as for the Integrate operator.

CONFIG Application for Validating Autonomous Control Software

CONFIG has been used for dynamic simulation-based test and validation of reactive sequencing software. CONFIG simulations were used to validate software that provided control during the Lunar Mars Life Support Test Program (LMLSTP) Phase III 90-day manned test, for Interchamber Monitoring and Control (IMC) of product gas transfer (PGT) between the crew chamber, plant growth chamber and an incinerator. The basic configuration of chambers for the test is shown in Figure 1.

The IMC PGT reactive task sequencer is the middle tier in the three-tiered (3T) autonomous control architecture (Bonasso et al. 1997). It is implemented in the Reactive Action Package (RAP) robot control language (Firby, 1997). The uppermost tier of 3T is a planner, which interfaces with the sequencer and can alter the sequencer's task agenda. The lowest tier of 3T is the skill manager layer that handles traditional low level control and

interfaces with both the sequencer and the LMSLTP Phase III testbed hardware. During simulation-based testing, the

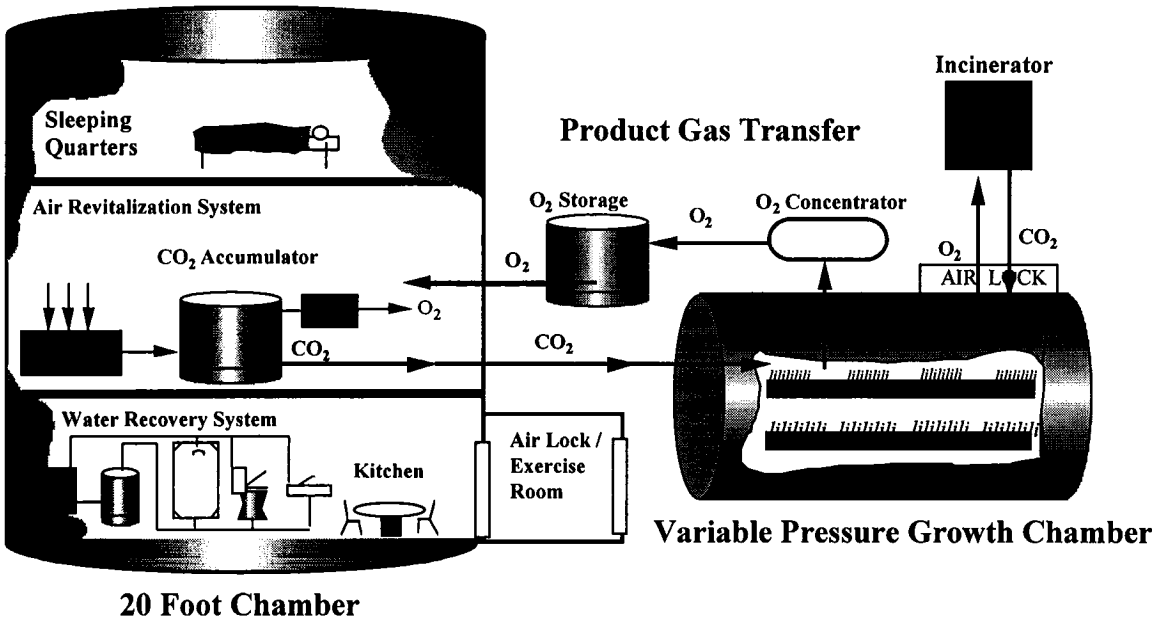

Figure 1. Product Gas Transfer in the Phase III Life Support Test

IMC sequencer software monitored and controlled the model rather than the skills layer and hardware. The model included diverse components and systems for processing O2 and CO2 gases in a plant growth chamber, crew chamber and incinerator, and for storing gases and transferring them between chambers (Fleming et al.1998a). Figure 2 shows the Product Gas Transfer system model during a simulation. The arrowheads along relations indicate the directions of active gas flows. The default graphic representations are rectangles for devices and elongated ovals for activities. Modes are indicated by the text in the rectangles and ovals, or by appearances of icons that indicate device modes.

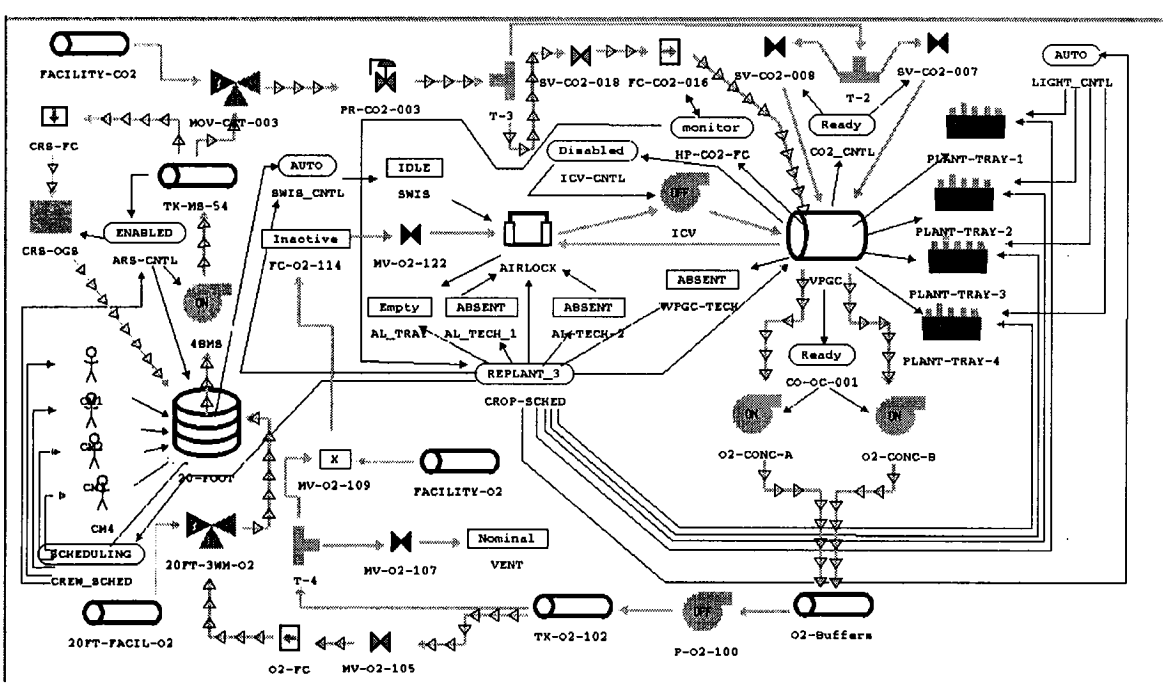

Figure 2. Product Gas Transfer System Model

The model devices include the chambers, various gas processors that convert oxygen to carbon dioxide or vice versa, gas concentrators, and PGT hardware that directs and regulates flow and pressure. The model activities include discrete and continuous control of the hardware that directs and regulates flow and pressure, schedules for crops and human activities, and some manual procedures. The activities model control by the 3T planning or skills tiers, local controllers or human operators. There are only two continuous feedback controllers in the 3T skills tier. The rest of the control is discrete, based on deadbands and schedules.

The simulation-based testing and its results are documented in Fleming et al. (1998b) and Malin et al. (1998). The testing uncovered some software bugs and some issues concerning software requirements. The most interesting issue was observed in the context of a complex interaction including elements of the crew chamber and the plant growth chamber. It is not likely that this type of software problem would have been found during conventional software testing because it involved a sequence of interactions of multiple devices and controllers in the system that would be difficult to conceive of or emulate in conventional software testing.

During simulation tests, when the CO2 accumulator was depleted the IMC software switched the source of CO2 from the accumulator to the facility supply as intended, except when the plant chamber CO2 level was between the alert-low and alarm-low thresholds. When the plant chamber CO2 level was below the alert-low level (1000 ppm) and the CO2 accumulator on the crew chamber side was also at its alert-low limit (12 psi), the IMC software failed to switch to the facility CO2 supply. The IMC software disabled continuous flow into the plant chamber and handed over control to the local CO2 controller in the plant chamber. The local controller then switched to the backup pulse injection system to raise the CO2 level in the plant chamber. Because the IMC software had failed to switch the CO2 source from the accumulator to the facility supply, the backup system attempted to draw CO2 from the depleted accumulator. The CO2 level in the plant chamber continued to drop even with the backup system on.

Future Work: Interaction of Discrete Simulators with Continuous Control Software

CONFIG has been used successfully and productively in validation of discrete control software. It would be desirable to extend its capabilities to test software that performs continuous process control functions, such as PID control. Two examples of such control were present in the Phase III Product Gas Transfer control software. One controller maintained the rate of carbon dioxide from an accumulator to the plant growth chamber so that the concentration of CO2 in the chamber air was maintained within close tolerances as the consumption rate of the plants varied. Another controller performed a similar function for supplying the plant chamber with carbon dioxide produced by incineration of waste. Rather than providing the plant chamber with CO2 from a pressurized source, this supply method involved circulating air between the airlock and the plant chamber. As the concentrations of CO2 in the two chambers equalized, the blower fan rate had to be increased by the controller. The CONFIG model contained "Activity" representations of both these control functions to support testing of the discrete control software. However, constructing these activities was in effect duplicating the work of the programmers who wrote the continuous control software.

It would be more useful if, in the future, CONFIG simulations could interact with such continuous process control software so that the simulation could provide a test platform for continuous as well as the discrete control software. It is anticipated, however, that there will be problems in interfacing continuous control software not encountered with discrete control software due to the incompatibilities between discretized representations of continuous processes and software designed to interact with truly continuous processes.

The concept of operating modes of devices is needed for validating reactive discrete sequencing software for control and fault management. Therefore, we expect that continuous-model-based hybrid simulation will be integrated within operating modes of devices. A change in an operating mode or transition to a failure mode can have both local and global effects. We have developed a simulation capability to handle global changes in flow existence and direction that can result from local mode changes. Otherwise, in CONFIG, complex behaviors can emerge from the interaction among devices. We anticipate problems with integrating hybrid continuous models that cross device boundaries. A global approach such as the one used to compute flow changes may be needed. We hope to discuss some of these issues at the workshop.

References

Bonasso , P., R.J. Firby, E. Gat, D. Kortenkamp, D. Miller and M. Slack. 1997. Experiences with an architecture for intelligent, reactive agents. *J. Experimental and Theoretical AI*, 9:237-256.

Firby, R. J. 1997. *The RAP Language Manual.* Neodesic Corporation.

Fleming, L., Hatfield, T. and Malin, J. 1998a. *Simulation-Based Test of Gas Transfer Control Software: CONFIG Model of Product Gas Transfer System.* Automation, Robotics and Simulation Division Report, AR&SD-98-017, NASA Johnson Space Center.

Fleming, L., Hatfield, T. and Malin, J. 1998b. *Simulation-Based Test of Gas Transfer Control Software: Software Validation Test Results.* Automation, Robotics and Simulation Division Report, AR&SD-98-018, NASA Johnson Space Center.

Malin, J. T.; Basham, B. D.; and Harris, R. A. 1990. Use of qualitative models in discrete event simulation for analysis of malfunctions in continuous processing systems. In Mavrovouniotis, M. ed., *Artificial Intelligence in Process Engineering.* San Diego, Calif.: Academic Press, 37-79.

Malin, J. T. , Fleming, L. and Hatfield, T. R. 1998. Interactive Simulation-Based Testing of Product Gas Transfer Integrated Monitoring and Control Software for the Lunar Mars Life Support Phase III Test. SAE Paper No. 981769. SAE 28th International Conference on Environmental Systems, Danvers MA.

Malin, J. T.; and Leifker, D. B. 1991. Functional modeling with goal-oriented activities for analysis of effects of failures on functions and operations. *Informatics & Telematics* 8(4):353-364.

Malin, J. T.; Ryan, D.; and Fleming, L. 1993. CONFIG - Integrated Engineering of Systems and their Operation. In *Proc. Fourth National Technology Transfer Conference*, 97-104. NASA Conference Publication CP-3249.

Malin, J. T.; Ryan, D.; and Fleming, L. 1994. Computer-aided operations engineering with integrated models of systems and operations. In *Proc. Dual Use Space Technology Transfer Conference and Exhibition*, 455-461. NASA Conference Publication CP-3263.

Zeigler, B. P. 1976. *Theory of Modeling and Simulation.* New York: Wiley.

Zeigler, B. 1998. Systems Theory Background for Continuous/Discrete Integration. SAE Paper No. 981767. SAE 28th International Conference on Environmental Systems, Danvers MA.

Zeigler, B. and Praehofer, H. 1998. Interfacing Continuous and Discrete Models for Simulation and Control. SAE Paper No. 981725. SAE 28th International Conference on Environmental Systems, Danvers MA.