'90 KACC 1990. 10. 26~27

MODEL-BASED DESIGN OF HIERARCHICAL EVENT-BASED CONTROL

Sung-Do Chi and Bernard P. Zeigler

AI-Simulation Research Group
Department of Electrical and Computer Engineering
University of Arizona
Tucson, AZ, U.S.A. 85721

ABSTRACT

Intelligent Control is an extended paradigm that subsumes both control and AI paradigms, each of which is limited by its own abstractions. Autonomy, as a design goal, offers an arena where both control and AI paradigms must be applied -- and a challenge to the viability of both as independent entities. We discuss hierarchical event-based control architectures in which AI and Control paradigms can be integrated within a model-based approach. In a model-based system, knowledge is encapsulated in the form of models at the various layers to support the predefined system objectives. Concepts are illustrated with a robot-managed space-borne chemical laboratory.

1. INTRODUCTION

Conceptions about how artificial intelligence fits into the world of systems engineering have been evolving rapidly. From expert systems applied in singular contexts, there emerged the concept of intelligent control [1]. As such application contexts as autonomous land and space vehicles, artificial worlds, telerobotics, and factories of the future multiply, it is becoming clear that autonomy, rather than intelligence, may be the more descriptive characterization of the new systems. While not minimizing the role of AI techniques, spelling out the engineering goals of higher autonomy might establish a less transient ladder of achievement than specifying the general, and nebulous, goal of building intelligent artifacts.

The work described in this paper derives from research related to telerobotics. The project is to develop a technology that will allow carefully designed and specially constructed laboratory robots to perform many routine tasks in a Space laboratory under remote supervision from the ground. Therefore the robots must be able to perform simple operations autonomously, and communicate with the ground only at the task level and above. Such robots should be able to judge the adequacy of a proposed action plan on the basis of expectations of its effects on the laboratory, materials, instruments, etc. For this purpose, it is important that models at various levels of granularity can be automatically generated from a set of generic master models[2].

The Systems Entity Structure/Model Base (SES/MB) framework was proposed by Zeigler as a step toward marrying the dynamic-based formalism of simulation with the symbolic formalisms of AI [3,4,5]. This knowledge/model base tool supports model-based hierarchical event-based control structures. The model base is a multi-level, multi-abstraction, and multi-formalism knowledge base and is kept coherent through the use of system morphisms to integrate related models[6,7,8].

Hierarchical models of components within the laboratory environment are constructed at different levels of abstraction. They have been implemented in DEVS-Scheme[3,5,9,10,11], a knowledge-based simulation environment for modelling and design that facilitates construction of families of hierarchical models in a form easily reusable by retrieval from a model base.

This paper is organized as follows. First it briefly shows the role of SES/MB framework in the model-based approach. Then it presents a System Entity Structure of a robot-managed laboratory. This is followed by the event-based control paradigm and multi-abstraction approach used to build a hierarchical event-based control structure. Finally we introduce a planning approach that is integrated with a hierarchical event-based control structure to generate autonomous laboratory systems.

2. THE ROLE OF SYSTEM ENTITY STRUCTURE/MODEL BASE FRAMEWORK IN MODEL-BASED ARCHITECTURE

The *System Entity Structure/Model Base (SES/MB)* framework was proposed by Zeigler[3,4,5] as a step toward marrying the dynamics-based formalism of simulation with the symbolic formalism of AI. It consists of two components: a *system entity structure* and *model base*. The *system entity structure*, declarative in character[9,10], represents knowledge of decomposition, component taxonomies, and coupling specification and constraints. The *model base* contains models which are procedural in character, expressed in dynamic and symbolic formalisms. The *entities* of entity structure refer to conceptual components of reality for which models may reside in the model base. Also associated with entities are slots for attribute knowledge representation. An entity may have several *aspects*, each denoting a decomposition and therefore having several entities. Associated with an aspect is *coupling* information needed to interconnect the entities of that aspect. An entity may also have several *specialization*, each representing a classification of the possible variants of the entity.

One application of the SES/MB framework is to the design of systems. Here the SES serves as a compact knowledge representation scheme for

organizing and generating the possible configurations of a system to be designed. To generate a candidate design we use a process called *pruning* which reduces the SES to a so-called *pruned entity structure*(PES). Such structures are derived from the governing structure by a process of selecting from alternatives where ever such choices are presented. Not all choice maybe selected independently. Once some alternatives are chosen, some options are closed and others are enabled. Moreover, rule may be associated with the entity structure which further reduce the set of configurations that must be considered. Initial planning in the model-based approach is based on the SES pruning process. This PES (initial plan) is in turn transformed into a simulation model (for execution). Thus the PES should be indexed to facilitate subsequent recognition and retrieval. As shown in Figure 1, PESs are stored along with the SES in files forming the *entity structure base*.

In DEVS-based simulation environment[3,5,9], hierarchical simulation models may be constructed by applying the *transform* function to pruned entity structures in working memory. As it traverse the pruned entity structure, *transform* calls upon a retrievals process to search for a model of the current entity. If one is found, it is used and transformation of the entity subtree is aborted. *Retrieve* looks for a model first in working memory. If no model is found in working memory, the *retrieve* procedure searches through model definition files, and finally, provided that the entity is a leaf, in pruned entity structure files. A new incarnation of the *transform* process is spawned to construct the leaf model in the last case. Once this construction is complete, the main *transform* process is resumed.

The result of a transformation is a model expressed in an underlying simulation language such as DEVS-Scheme[10] which is ready to be simulated and evaluated relative to the modeler's objective. The fact that the *transform* process can look for previously developed pruned entity structures, in addition to basic model files, encourages PES reusability.

In model-based design approach, knowledge is encapsulated in the form of models that are employed at the various control layers to support the predefined system objectives. Lower layers are more likely to employ conventional differential equation models with symbolic models more prevalent at higher layers. A key requirement is the systematic development and integration of dynamic and symbolic models at the different layers. In this way, traditional control theory, where it is applicable, can be interfaced with AI techniques, where they are necessary. Discrete event representation, facilitating event-based control, can be employed to map traditional dynamic to symbolic models[5,6].

Note than an autonomous system could in principle, base its operation, diagnosis, repair, planning, and other activities on a single comprehensive model of its environment. However, such a model would be extremely unwieldy to develop and lead to intractable computations in practice[2]. Instead, our architecture employs a multiplicity of partial models to support system objectives. As indicated, such models differ in level of abstraction and in formalism. The partial models, being oriented to specific objectives, should be easier to develop and computationally tractable[3]. However, this approach leads to sets of overlapping and redundant representations. Concepts and tools are need to organize such representation into a coherent whole. Morphisms[3,4,12] can connect models at different levels of abstraction so that they can consistently modified.

Fishwick[13,14] has extended process abstraction concepts and implemented a simulation system that is able to switch between levels within simulation runs. However, although the need for multiple levels of abstraction has been recognized in mainstream AI, there has been little consideration of the importance of the morphism concept to this issue. But just as tools are needed to enable agents to plan, diagnose, and reason effectively with respect to particular objects, so tools are needed to organize the various models that support such planning, diagnosis, and reasoning. An organized model base enables the agent to deal with the multiplicity of objects and situations in its environment and to link its high level plans with its actual low level actions. Such a model base is a special case of multifacetted model base management[3].

The SES/MB framework provides an ability to develop model-based planning systems. It can supports:

(1) multiplicity of partial models to support system objectives (multifacetted modelling).
(2) integration of dynamic and symbolic models at different layers (hierarchical architecture).
(3) multi-abstraction to integrate related models (system morphism).
(4) execution, control, diagnosis, and repair (event-based control).
(5) selection/retrieval of initial/modified planning models (pruning and reusability).

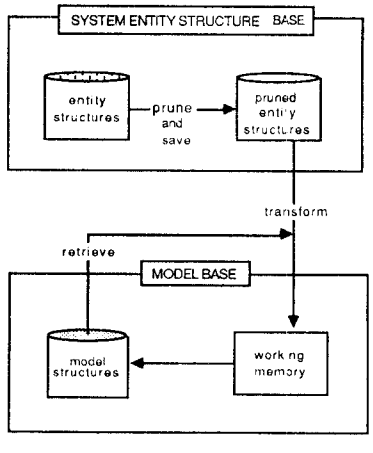

Figure 1. The System Entity Structure/Model Base (SES/MB) Environment

3. EXAMPLE : SPACE-BORNE LABORATORY

As an example of a model-based architecture for hierarchical event-based control we consider a space-borne, robot-managed laboratory environment for chemical, biological and other scientific experiments upon the planned Space Station.

It is timely to begin exploration of advanced robot-controlled instrumentation. For example, handling fluids in orbit will be essential to many of the experiments being planned in manufacturing and biotechnology. However, the micro gravity conditions of space necessitate radically different approaches to fluid handling than common on earth. As experience in space accumulates, approaches and instrumentation will likely undergo continual modification, enhancement, and replacement. Thus robots for managing such equipment must be highly autonomous and flexible so that constantly changing environments can be accommodated.

In designing the robot models, we assume that necessary mobility, manipulative and sensory capabilities exist so that we can focus on task-related cognitive requirements. Such capacities, the focus of much current robot research, are treated at a high level of abstraction obviating the need to solve current technological problems.

The laboratory environment is constructed on the basis of object-oriented and hierarchical models of laboratory components within DEVS-Scheme [5]. Laboratory configurations will be determined by issuing natural language commands which initiate a pruning operation of the entity structure knowledge representation. The laboratory model is designed to be as generic as possible. However, as stated, the focus will be upon fluid handling in microgravity which presents a variety of problems that are unique to space.

Figure 2 illustrates the approach taken. The entity structure for SSL (Space Station Laboratory) decomposes this entity into a structure knowledge part and experiment (goal) knowledge part: STRUCTURE and EXPERIMENT. The former specifies how to construct the autonomous system, and latter concerns how to achieve given experimental goals. Each of the entities will have one or more classes of objects (models) expressed in DEVS-Scheme to realize it.

The EXPERIMENT has a three level abstraction hierarchy: high-level model (HM), middle-level model (MM), and low-level model (LM), which are associated with Model-Plan Units (cognitive control elements), MPUs; HMPU, MMPU, and LMPU in STRUCTURE part, respectively. Each level is designed to represent experimental knowledge needed to decompose a given task into a composition of subtasks.

The SPACE and OBJECTS decomposing STRUCTURE are designed for the simulation-oriented knowledge representation, which consist of *controlled model* within DEVS environment. The SPACE is a controller and OBJECTS are controlees in a *"controlled model"* structure [5].

Each OBJECT is specialized into ROBOT and EQUIP. And each ROBOT is decomposed into MOTION, SENSE, and BRAIN. EQUIP is a generic entity for laboratory equipment and is modeled much the same as the ROBOT. However, EQUIP has no BRAIN and its MOTION and SENSE subsystems are always passive (we are considering "dumb" equipment here). Note that OBJECTS is defined as a *multiple entity* (any number of instances may be generated for such an entity), and with the pruning discussed earlier, we can have any desired number of ROBOTs and EQUIPs in the laboratory.

Each Robot's Cognition system (BRAIN) is also a controlled model containing a SELECTOR as controller, and MPUs (Model-Plan Units) as components. MPU is specialized into HMPU, MMPU, and LMPU corresponding to the EXPERIMENT abstraction hierarchy just mentioned. HMPU is a high level MPU which manages the actions of the next level MMPUs, each of which perform the same function with respect to the lower model LMPUs. The latter employ event-based control logic to interact with the environment. These three types of MPU are represented at the same level in our entity structure but conceptually they reflect the hierarchical decomposition structure of the EXPERIMENT models.

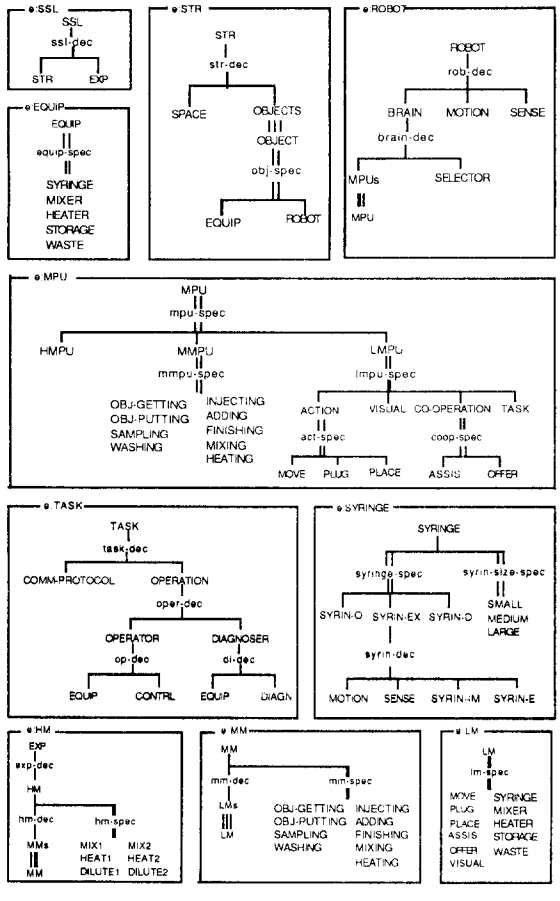

Figure 2. Partitioned SES of Robot-Managed Chemical Laboratory

4. EVENT-BASED CONTROL

DEVS (Discrete Event System Specification) [6,11] models of equipment can be used to guide robotic manipulation of devices in the execution of typical laboratory procedures such as filling, mixing, heating, etc.

In the conventional approach to control, the controller sends out a command to the data acquisition sub-system to sample the process at regular intervals. When the sampled value returns, it is stored and tested. Depending on the outcome of the test, a corrective control action command is emitted. Testing of the sampled value is performed by determining whether it lies within a window, i.e., a sub-interval of the sensor output range.

The alternative form of control logic, called *event-based control*, was introduced by Zeigler [6,11]. In this control paradigm, the controller expects to receive confirming sensor responses to its control commands within definite time windows determined by its model of the system under control. This control unit is consisted of a *planner*, *controller*, and *diagnoser*, and internal models derived from the lower-level part of the EXPERIMENT abstraction hierarchy as shown in Figure 3.

The *planner* works by developing paths backward from the goal until the given initial states (possible starting states of the controlled system) are reached [5].

The event-based *operator*, consists of the two components, a *controller*, DEVS internal model. Such an operator is one sub-component of a LMPU within a robot cognition model. The controller is a generic engine, similar to an inference engine in expert systems, which uses the internal model, LM (e.g. MI-O). The controller obtains information from the model relating to command, and expected responses times and windows. Then, it issues these commands to its controlled device, ME. If the model is valid, and operation proceeds normally, the underlying homomorphic relation (discussed later in detail) is maintained between the model and the controlled device. The controller ceases interacting with the device as soon as any discrepancy occurs in this relationship and calls on a diagnoser to figure out what happened.

Once the controller has detected a sensor response discrepancy, the *diagnoser* is activated. Data associated with the discrepancy are passed on to the diagnoser. From such data, the diagnoser tries to discover the fault that occured by using its diagnostic model, (e.g., MI-D).

An essential advantage of event-based control is that the error messages it issues can bear important information for diagnostic purposes. This possibility aries when a DEVS model is developed for the process and used to determine the time windows for sensor feedback. As a side benefit, causes for other-than-expected responses may also be deduced.

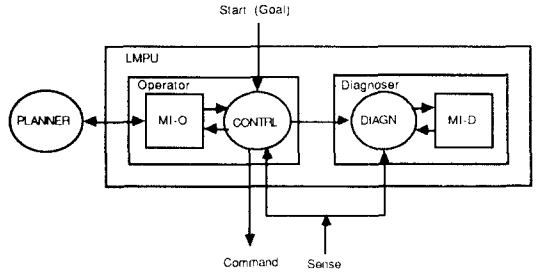

Figure 3. Event-based Control Structure

5. MODEL ABSTRACTION HIERARCHIES

Models of intelligent agents must represent not only a decision making component, but also the model of the real system the decision maker uses to arrive at its decisions. Such modelling is based on homomorphic preservation of the input-output behavior where inputs are operation commands to the system and outputs are responses of finite-state sensors attached to the system to observe its state. Selection of controls and sensors must reflect the operation objectives. An atomic DEVS model abstracts incremental micro-state transitions from the continuous model and replaces them by time windows taken for macro-state transitions (which correspond to crossing of sensor thresholds). A coupled DEVS model abstracts the behavior of the composition of lower level DEVS models.

Figure 4 represents abstraction related models. At the bottom, MB is a continuous state model of the system being controlled (the most refined model considered for it). Other models are related by abstraction, i.e., a form of homomorphic relation. ME is a discrete event model derived from MB, and MI-O and MI-D are two different abstraction models of MI (for operation and diagnosis, respectively) which in turn is an abstraction of ME. Each abstraction is governed by an underlying morphism. ME serves as the external model of each device, whereas MI-O and MI-D serve as the internal models used by the low level event-based control units (LMPU).

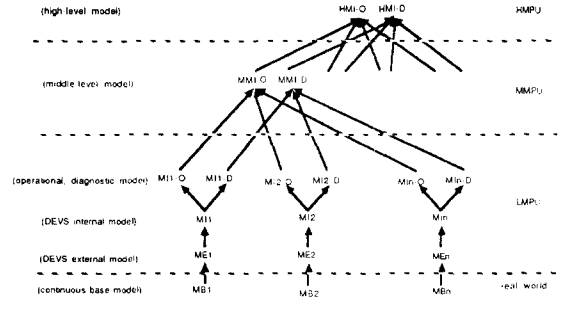

Figure 4. Abstraction Related Models

Sets of MI-O and MI-D can be composed into higher level models (MMI-O and MMI-D) and abstracted to represent more global state transitions (used by MMPU). Likewise, HMI-O and HMI-D are the highest level models which represent global state transitions (used by HMPU). In this way, the

higher level execution units use their own models, which are abstractions of compositions of lower level models, to control their subordinates. The abstraction process is done by aggregating states and windows of lower level models.

The levels of abstraction just illustrated can be formalized using system morphism concepts[3,5]. By generating a hierarchical structure with models of different levels of abstraction we can systematically integrate the highest level goal command with most refined dynamic models.

6. HIERARCHICAL EVENT-BASED CONTROL

As stated earlier, the robot's cognition system exemplifies *hierarchical event-based control*. This structure is derived from the EXPERIMENT model abstraction hierarchy in which a higher-level model unit uses its associated model to supervise lower-level model units.

There are three levels of Model-Plan Units: high-level (HMPU), middle-level (MMPU), and lowest-level (LMPU).

The HMPU first formulates a given task (e.g., command from earth based scientist) and then divides it into action-oriented sub-plan units, MMPUs. Here actions are sequenced (planned) by chaining necessary actions in a combined forward/backward manner starting from the given target action which is assigned by the task formulation process.

The MMPUs in the hierarchy are decomposed again into LMPUs. To control its lower level units, MMPU has abstracted states and time windows of its lower level LMPUs.

The LMPU is the lowest level in the hierarchy which employs the event-based control logic for operation and fault diagnosis.

The hierarchical event-based control structure is shown in Figure 5. At each level, the control unit has its own internal model and controller to supervise its sub-component units. There are two types of messages in the hierarchy: goal (command) and done (response) messages. The goal is divided into a set of subgoals in top-down manner, whereas, the *done messages* are gathered in bottom-up manner. There are three types (+,0,-) of *done messages*: + is for a *success*, - for a *fail*, and 0 for an *unknown*. The unknown message may be due to the lack of available sensors or complex fault associated with other units in the hierarchy.

As a concrete illustration, let us set up a mixing experiment, i.e., *"mix x amount of liquid-A with y amount of liquid-B"* in the space-borne laboratory environment. To perform such a task, a robot must identify a syringe required to sample a liquid-A, then bring that syringe to the identified storage in which liquid-A is stored, and perform the sampling from the storage, and so on. As shown in Figure 6, the HMPU manages such activation sequences (MMPUs), with each action unit (MMPU) divided into smallest action units (LMPUs).

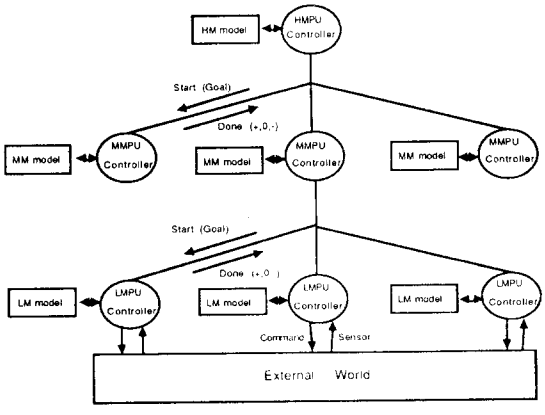

Figure 5. Hierarchical Event-Based Control Structure

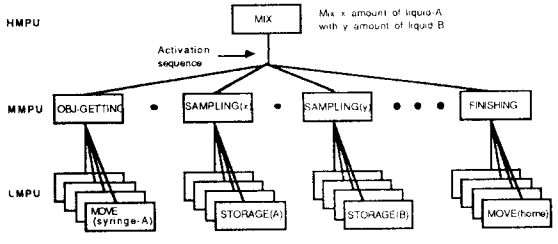

Figure 6. Mixing Example of Hierarchical Event-based Control Logic

7. AUTONOMOUS SYSTEM GENERATION

Having developed DEVS model base and constructed the system entity structure supplies the framework to generate a control structure with a given goal. In other words, once we have specified the available resources (structures) and their behavior characteristics (models), we have to specify how to plan, i.e., allocate resources to accomplish a goal. In our SES/MB framework approach, the planning is viewed as a pruning operation which generates a candidate structure. As shown in Figure 2, the STRUCTURE entity represents an execution structure which has the possible configurations of a system to be pruned. On the other hand, the EXPERIMENT entity represents a goal structure which has the possible alternative goals. The execution structure is pruned by the pruned goal structure. By issuing a command, the system should autonomously: interprete command, plan the activation sequence, select models, execute the plan using control, diagnosis, and replanning to assure eventual successful execution. Figure 7 depicts a model-based methodology for generating autonomous system as follows ;

1) Load the system entity structure (SSL) which organizes all available domain knowledge including experiment knowledge and structure knowledge;
2) Interpret the natural language stated task goal to map into the predefined domain knowledge (SSL);
3) Check the entity structure base (ENBASE) to see whether there are already existing plans (pruned

entity structures); if not, select necessary models and sequence actions by pruning to the goal knowledge part of the system entity structure (EXPERIMENT);

4) Construct a model structure by pruning the execution knowledge part of the system entity structure (STRUCTURE);
5) Transform the model structure into an autonomous system architecture by synthesizing component models from the model base, MBASE.
6) In the fault case, reprune (replan) the SES by issuing the repairing goal and repeat from step (3).
7) Save the results for reuse; states of each model into MBASE and PES (pruned entity structure) into ENBASE.

The reuse of existing plans formulated as pruned entity structures is similar in spirit to the case-based planning approach [15,16].

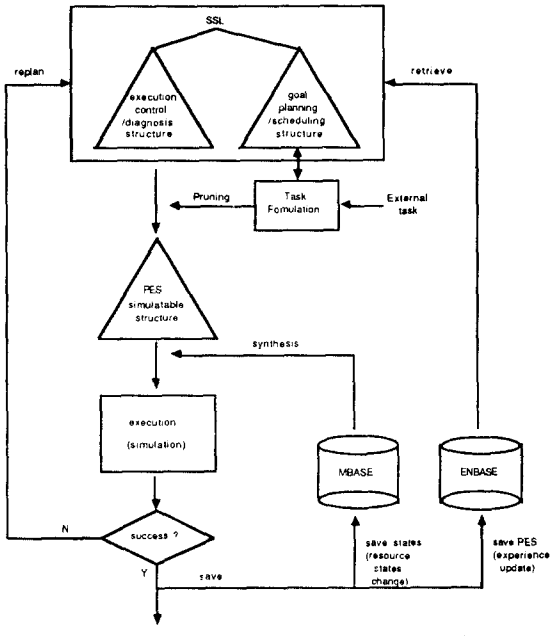

Figure 7. Autonomous System Generation Methodology Using SES/MB

8. CONCLUSIONS

Building on the basis of the SES/MB framework as implemented in DEVS-Scheme, we have extended the ability of our knowledge/model base tools to support model-based design of hierarchical event-based control structure. We have developed the methodology of the autonomous system generation by integrating the execution structure (resources structure) and planning structure (resources allocation structure).

As described above, the model-based approach, employing multi-abstraction, multi-level, and event-based control logic, has been demonstrated in the design of a robot-managed space-borne laboratory environment. We have applied our autonomous system to fluid handling under microgravity conditions. Although much of the methodology has been implemented, much work remains to complete and verify the working system.

REFERENCES

[1] A. Meystel, "Intelligent Control: A Sketch of the Theory," *J. Intelligent and Robotic Systems*, vol.2, No.2&3, pp.97-107, 1989.

[2] Q.Wang and F.E.Cellier, "Time windows : An Approach to automated Abstraction of Continuous-time Models into discrete-Event Models", *Proc. on AI, simulation and Planning in High Autonomy systems*, Tucson, March, 1990.

[3] B.P. Zeigler, *Multifaceted Modelling and discrete Event simulation*, Academic press, 1984.

[4] B.P. Zeigler, "Knowledge Representation from Minsky to Newton and beyond", *Applied Artificial Intelligence*, vol.1, pp87-107, Hemisphere Pub. Co. 1987.

[5] B.P. Zeigler, *Object-Oriented simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic systems*, Academic Press, 1990.

[6] S.D. Chi and B.P. Zeigler, "DEVS-based intelligent Control of Space Adapted Fluid Mixing", *Proc. of 5th conf. on Artificial Intelligence for Space Applications*, May, 1990.

[7] B.P. Zeigler and S.D. Chi, "Model-based Architecture Concepts for Autonomous Systems", *Proc. on 5th IEEE Int. Symposium on Intelligent Control*, 1990 (in process).

[8] B.P. Zeigler and S.D. Chi, "Hierarchical Systems Architecture for Artificial Intelligence", *Proc. on 34th Int. Soc. System Sciences Conf.*, Portland, July, 1990.

[9] T.G. Kim, "A knowledge-based Environment for Hierarchical Modelling and Simulation", Ph.D Dissertation, Univ. of Arizona, 1988.

[10] T.G. Kim, and B.P. Zeigler, ESP-Scheme : A realization of system Entity Structure in a LISP Environment," *Proc. in 1989 SCS Eastern Multiconference*, March 1989, Tempa, Florida.

[11] B.P. Zeigler, "DEVS Representation of Dynamical Systems: Event-Based Intelligent Control", *IEEE proc.* Vol.77, no.1, . pp.72-80, Jan., 1989.

[12] B.P. Zeigler, *Theory of Modelling and Simulation*, Wiley, NY, (Reissued by Krieger Pub. Co., Malabar, FL, 1985), 1976.

[13] P.A. Fishwick, "A Taxonomy for Process Abstraction in Simulation Modelling", *IEEE Int. Conf. Sys. Man & Cyb.*, Vol. 1, pp. 144 - 151, 1987.

[14] P.A. Fishwick, "Abstraction Level Traversal in Hierarchical Modelling", In: *Modelling and Simulation Methodology: Knowledge Systems Paradigms* (Eds.: M.S. Elzas, T.I. Oren, and B.P. Zeigler), North Holland Pub. Co., Amsterdam, pp. 393 - 430, 1989.

[15] K.J. Hammond, *Case-Based Planning*, Academic Press, 1989.

[16] C.K. Riesbeck and R.C. Schank, *Inside Case-Based Reasoning*, Lawrence Erlbaum Associates, Pub., Hillsdale, NJ, 1989.