An Object-Oriented Multimodel Approach to Integrate Planning, Intelligent Control and Simulation

Jin Joo Lee William Dean Norris II Paul A. Fishwick

Department of Computer and Information Sciences
University of Florida
Gainesville, FL 32611

Abstract

The areas of planning, intelligent control and simulation have each spawned their own representational structures. Deliberative planning approaches found within AI are often rule based and simulation is often function based. Intelligent control approaches and the perceived need to integrate reactive control with deliberative planning has suggested an integration of modeling techniques; however, well-formed integration techniques are scarce. We propose using the multimodel method for large scale systems containing intelligent and non-intelligent objects. Our initial results with planning and control within a truck depot show that, with multimodeling, current deliberative planning methods are extended to include a richer set of model types besides production rule systems. Moreover, reactive control is modulated by deliberative planning model structures so that the entire model is well-formed and easy to comprehend. **[Key Words: Autonomy, Intelligent Control, Multimodeling, Planning, Object Oriented Simulation]**

1 Introduction

The object oriented approach to simulation is discussed in different literature camps. Within computer simulation, the system entity structure (SES) [20] (an extension of DEVS [17, 16]) defines a way of organizing models within an inheritance hierarchy. In SES, models are refined into individual blocks that contain external and internal transition functions. Within the object oriented design literature [18, 1], the effort is very similar in that object oriented simulation is accomplished by building 1) a class model and 2) dynamic models for each object containing state information. Harel [10, 11] defines useful visual modeling methods in the form of "state charts" so that the dynamics may be seen in the form of finite state machines. From our perspective the object oriented approach provides an excellent starting point when deciding how to organize information about dynamical systems:

1. Start with a concept model of the system.
2. Create a class model using a visual approach such as OMT [18]. This phase should involve creating all relationships among classes.
3. Specify the dynamics for each class instance where state transition is a factor. Note that some classes will not contain state information and some relations may not be of a dynamic nature.
4. Construct a *multimodel* to build a network of models each of which defines a part of the overall system.

In the usual object oriented approach, phase three translates to creating methods for an object that alter the state of that object. The problem is that phase three can be quite complex depending on the scale of the system being modeled. There needs to be a way of developing multi-level models that specify the phase three dynamics. Our approach is to use *multimodels* [8, 4, 5, 6, 13] for this purpose. Multimodeling is a paradigm for designing and executing models. We use several well defined model types and connect them, so that the lower levels refine the higher levels. Due

to the hierarchical structure of the multimodel [4, 5] approach, the object oriented paradigm is natural for implementation. Each of the model types are executed using the same methods, *Initialize()*, *Input()*, *Output()*, *State()* and *Update()*. Therefore, by having a method of communication between each model type, the models are executed regardless of which model types are used.

2 Integrating Simulation and Planning

The idea of integrated simulation and planning developed from our efforts to overcome the many problems inherent in artificial intelligence (AI) and simulation. Traditionally in AI planning, researchers are concerned in building a planner that produces a symbolic plan (order of primitive actions) which will achieve certain tasks. However, the planner is unable to control the execution or modify the plan in order to guarantee success of the proposed plan. Traditional AI planners were therefore unrealistic. Basically, these *deliberative* planners assumed perfect knowledge of a very static world. Taking the other extreme, other researchers developed purely *reactive* planners that can react to the environment by executing actions without extensive reasoning.

The next logical step is to combine planning and control to produce a planner that is both deliberative and reactive. Dean [3] provides a good overview of the various problems and techniques available in these two areas. Most of the traditional planners which have been built so far are either purely deliberative or purely reactive. Recently, there have been some efforts to develop a *combined* planner [12, 2, 9]. Due to the divided research between deliberative and reactive planners, the technology of the two fields has also been divided. We believe the major difficulty in trying to build a *combined* planner is integrating the different methods of each area. Thus, multimodeling allows the integration of these different techniques as submodels. With planning and control combined, we now want to integrate the different modeling types that exist in AI and simulation. Some previous work [14, 15] has been done in the integration of AI and Simulation. The integration has two major advantages. Because both the intelligent and non-intelligent objects are being modeled and simulated under one simulation, we are able to test and evaluate the performance of the overall system. From the planning perspective, testing, evaluation and modification can be done without connecting the planner to an actual physical device or object.

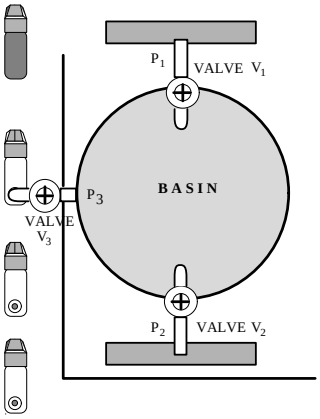

Figure 1: Concept model of truck depot (Aerial view)

The Truck Depot problem was originally taken from [3]. Since the problem contains both non-intelligent objects (e.g. basin, trucks, valves) and intelligent objects (e.g. robots or people) in equal emphasis, the problem inspired us to find a solution to optimization by combining the two fields of simulation (simulating non-intelligent objects) and AI (simulating intelligent objects) under a unifying modeling paradigm.

3 A Truck Depot Example

Fig. 1 shows the aerial view of the truck depot, which represents the concept model of system. The depot contains one basin with two input pipes P_1,P_2 carrying two different chemicals and one output pipe P_3. Each pipe has a valve (V_1, V_2 and V_3) which controls the flow. Each valve has a servo motor attached enabling the human operator to remotely close or open the valves. Empty tanker trucks arrive at the depot and wait until they can move under pipe P_3 to be filled with the mixture from the basin. When each truck leaves the basin, its cargo is tested. If the truck has been filled with an acceptable mixture, it leaves the system; otherwise it dumps the cargo and returns to be refilled. All mixture which overflows from the basin or the truck is treated as waste. In our version of the problem, the capacity of each tanker trucks is constant.

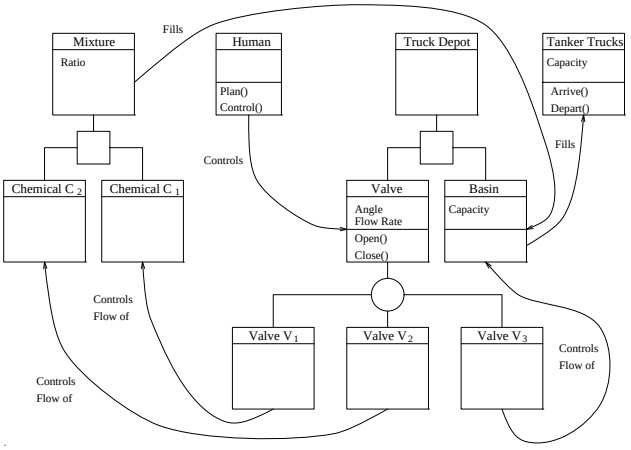

Figure 2: Instance model of truck depot

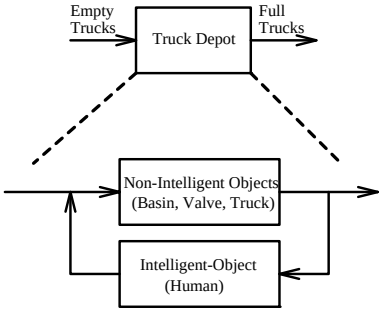

Figure 3: Top view of system

The intelligent objects control the non-intelligent objects (see Fig. 2). In our case, the human operator plans and controls the valves in the depot, while achieving the goal of maximizing the profit of the depot by maximizing the number of trucks filled while minimizing the cost. The depot is charged for the total amount of chemicals that flow through the input pipes during the period in which it is open. The trucks are independent objects which arrive according to an exponential distribution over the period of time when the depot is open.

We also have the notion of time to consider in our simulation. The start of simulation time corresponds to an opening time of the depot and the end of simulation time corresponds to the closing time in the real world. Thus, any remaining mixture in the basin after closing time will also be waste. Fig. 3 shows the control system of our simulation.

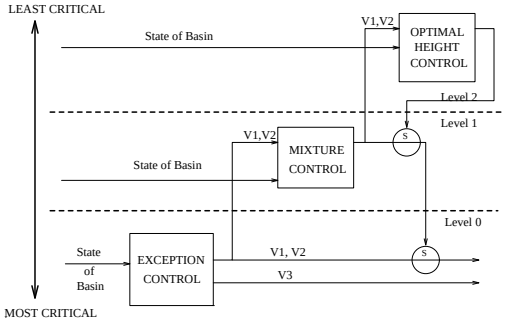

Figure 4: Multimodel planner

4 Intelligent Objects

Because the arrival of the trucks is dynamic and there is no determined number of trucks *a priori*, no type of offline planning is possible. As illustrated in Fig. 4, we have adopted Brook's subsumption architecture [2] to integrate the different level modules. Because our simulation is discrete, we will allow the output to be suppressed by a higher level for one time step until the next event arrives and causes another output. The multimodel planner has multiple levels that are divided based on how critical the reaction of each level is to the overall success of the planner. The levels also reflect the reactivity of the control in that the lowest level module is the most reactive module whereas the highest level is the least reactive. In general, however, this may not always be the case.

4.1 Exception Control

At the lowest level in the hierarchy the Exception Control module exists, which is the most reactive and the most critical. In our problem, there are two critical situations. First, overflow either from the basin or the truck must be avoided at any expense, since spilled mixture can never be recovered. Second, unless it is close to the end of simulation time, the planner should avoid having an empty basin since no truck can be filled. The Exception Control takes the state of the basin as input, which is the volume of the mixture in the basin, B_{vol} and the volume of the mixture in the truck, T_{vol}. With B_{vol}, fuzzy logic [19] is used to infer whether the basin is in an OVERFLOW or EMPTY state. With T_{vol}, fuzzy logic is used to decide if the truck is in an OVERFLOW state. Only when these conditions arise, does Exception Control

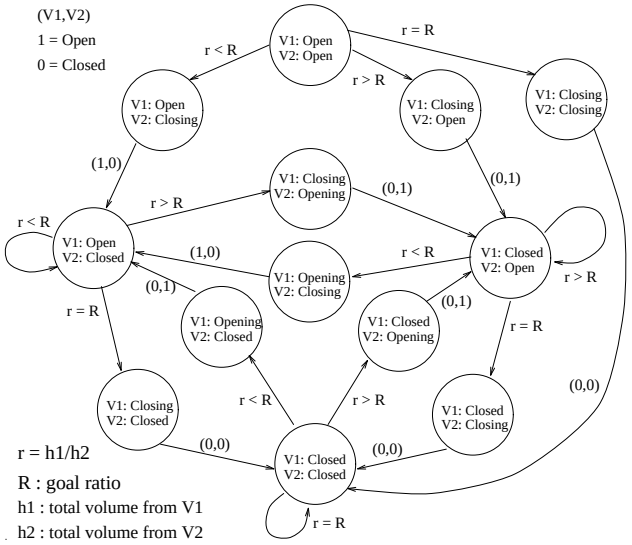

Figure 5: FSA for Mixture Control

react and send an output command (CLOSE or OPEN valve V_n). NO-CHANGE —which corresponds to a null signal— is sent otherwise.

At this level, the output fuzzy set is identical for both valves V_1 and V_2. Because when an overflow (or empty) occurs or is about to occur, both of the valves need to be closed (or opened) at the same time. To control the overflow of the truck, the volume of the mixture in the truck is monitored and inferred by fuzzy logic to be either in the state NOT_FULL or FULL. The output set for valve V_3 is similar to the one for valves V_1 and V_2. Fuzzy logic seems well suited since we are not concerned precisely when these events occur, but rather the appropriate time to start monitoring and controlling to prevent overflow. The fuzzy logic model best mimics the actual performance of the human operator at this level.

4.2 Mixture Control

The Mixture Control module is responsible for maintaining the *correct* ratio of the two chemicals in the basin. Initially, the planner is given a ratio R which is to be maintained throughout the simulation. The mixture is considered acceptable and to be of the *correct* ratio if the actual ratio r falls between the range $R-5\%$ and $R+5\%$. The type of model used for Mixture Control is a finite state automaton as shown in Fig. 5. In the figure, the current state of the valves is represented by the tuple (V_1, V_2) where V_n can be one of the following Open, Closed, Opening, Closing. Depending on the current state, the next state is reached by choosing the appropriate transition. If the next state reached is a transition state (state that contains one or more valve settings ending with *ing*), an output command is sent that will actually create the event to change the physical state of the valves. However, the output command may be suppressed by the higher level, therefore never reaching the basin model. This may also be the case for the Exception Control module. For more detailed explanation of the suppressor function, refer to [2].

4.3 Optimal Height Control

Finally, the Optimal Height Control module controls the height in order to maximize the profit. Because this module is less reactive and involves more symbolic knowledge and reasoning (heuristics) than the other lower level modules, rule-based reasoning is best suited for the task. The notion of optimal height is time dependent. Since the simulation has a start and an end time, the intelligent object can have different strategies for maintaining the height at different times. Another consideration that Optimal Height Control module takes into account is the speed of chemical flow. The flow rate of valve V_3 depends on the height of the mixture in the basin. Since our Optimal Height Control module uses heuristics, optimality is not guaranteed. Included in this module, is the Evaluator which evaluates the overall profit of the system, during and after the end of simulation. The formula used is $Profit = N(V_{gt}) - C(V_i)$ where N is the amount of reward per unit of volume and C is the amount of money charged per unit of volume. V_{gt} represents the total volume of good trucks and V_i represents the total volume of input.

5 Simulating Multimodels

5.1 What are Multimodels?

Models that are composed of other models, in a network or graph, are called multimodels [4, 5, 7, 6, 8]. Multimodels allow the modeling of large scale systems at varying levels of *abstraction*. They combine the expressive power of several well known modeling types

such as FSAs, Petri nets, block models, differential equations, and queuing models. By using well known models and the principle of orthogonality we avoid creating a *new* modeling system with a unique syntax. When the model is being executed at the highest level of abstraction, the lowest level (representing the most refined parts of the model) is also being executed. Each high level state duration is calculated by executing the refined levels that have a finer granularity.

5.2 Why use Multimodels?

For the Truck Depot example the question arises, "*Why use multimodels?*" The non-intelligent objects in the system could be modeled using control functions since classical control theory would provide an optimal solution. Therefore, using a multimodel may not seem necessary. However, multimodels offer several advantages over classical control theory.

- *Extensibility*: A multimodel is able to be extended. Making a change to the model, (e.g. adding evaporation of the mixture while in the basin or having a variable filling capacity for the tanker trucks) is difficult, if not impossible, to implement when using classical control theory.
- *Replaceability*: Any of the objects in the system can be replaced by another object that accepts the same input and gives the same output. For example, the two input valves can be replaced by three input valves.
- *Reusability*: The planner, the basin or the *entire* truck depot could be used within the context of a much larger model containing the depot as a component.
- *Comprehensibility*: Any physical system can be modeled using classical control theory, but it has drawbacks. For example, the equations for large scale systems become prohibitively complex and unsolvable when small changes are made to the system.

5.3 How to use Multimodels

Each type of model (e.g. Petri net, block, differential equation, FSA, queuing) has similar features: *input*, *output*, *state*, and a *transition* from one state to the next. Some model types have lower level components that encode information about the model. In an FSA, each state holds all of the information that is needed to answer any question about the model. In contrast, a transition and place in a Petri net each only have information about a subset of the system, although when combined, they describe the complete system.

To simulate a multimodel, it is necessary to have the input from one model type accept the output from another type of model. Each model must be able to recognize the output of any refining model as *valid* input.

To synchronize the system at its highest level, a coordinator is used to process the external inputs. The coordinator also creates and initializes each model and its components, and then organizes the models into the specified hierarchy. The coordinator executes the refining models, but allows only external output from the levels above the specified level. The coordinator uses a future event list (FEL) to keep track of: 1) the next event, 2) to which model or model component the event should be sent, 3) which token caused the event, and 4) the global time of the simulation. Each level of the model must wait for an event to begin execution, then it posts its new state to the FEL.

The following methods are used to simulate each of the model types. *ReadModel()* reads the model specifics and creates the model instance. *Initialize()* describes how the model and its refining models are initially set. *Input()* collects all of the input data that a model receives during execution. *State()* returns the current system state. *Update()* causes the input or event to be applied to the current state and the next state to be scheduled on the FEL. *Output()* returns the output of the current state.

6 Non-Intelligent Objects

6.1 Model Design

Modeling the basin poses several challenges. The model state includes continuous and discrete variables, constraints, and functional relationships.

The volume of the mixture in the basin changes continuously throughout the simulation. The input signals that control the three valves give continuous outputs, but they change at discrete times. The tanker

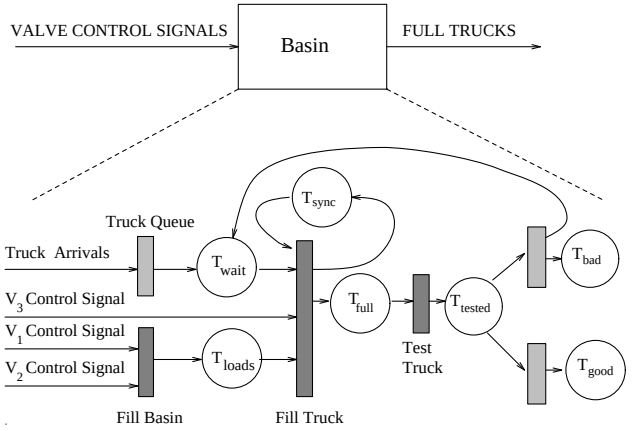

Figure 6: Petri net model of the basin

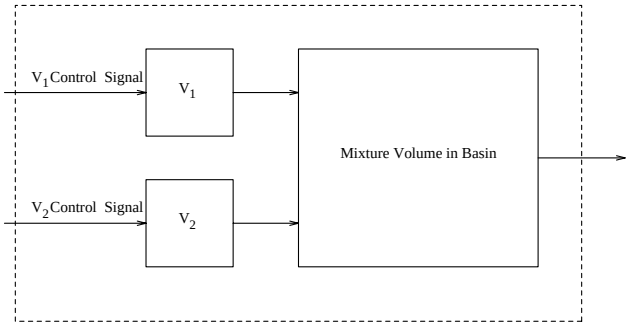

Figure 7: Refinement of transition *Fill Basin*

trucks move through the system as discrete objects, and are constrained when waiting to be filled because: the signal for valve V_3 must open the valve; the basin must have enough mixture to fill a truck; and the filling area must be empty.

We chose a Petri net to model the top level, see Fig. 6. The inputs to the model are tanker truck arrivals and control signals for opening and closing each of the valves. The output from the model includes statistics showing the number of trucks that were filled properly, and the volume of each chemical that was poured into the basin during the simulation.

The *Truck Queue* transition is refined by an S/S/1 queuing model. It was chosen to model the tanker trucks waiting to be filled. The queue is used to maintain the trucks arrival order.

The transition *Fill Basin* is refined by a block model, see Fig. 7. The control signals are passed to the refining models *Valve 1* and *Valve 2* respectively.

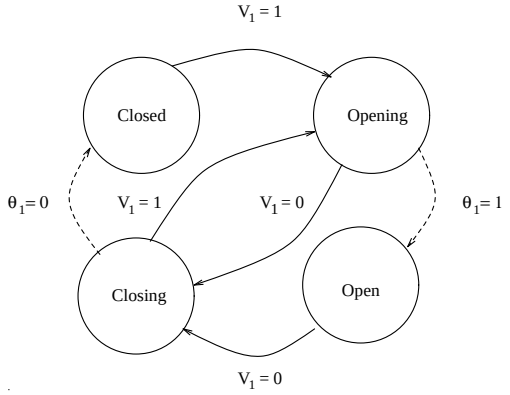

Figure 8: FSA model of valve V_1

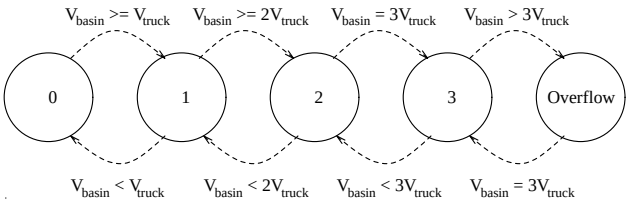

Figure 9: FSA model of the volume of mixture in the basin

Each refining model returns a value representing the control to be applied. The function block, *Mixture Volume in Basin* takes the control input and returns a value that states how many tanker truck loads the basin currently holds.

Fill Truck is also refined by a block model (similar to Fig. 7) that shows the relationship between valve V_3 and the *Mixture Volume in Truck* function block. Valve V_3 takes as input the control signals, *OPEN_V3* and *CLOSE_V3*, then returns the amount of control to be applied due to the current state of valve V_3. The function block *Fill Truck* takes the control value as input and returns a value when the truck has been filled.

Each of the function blocks are refined by a finite state automaton. There are five FSAs, but the three FSAs that refine the valves are similar to the refinement of valve V_1.

The function block *Valve* V_1 is refined by the FSA shown in Fig. 8. The input for this FSA is the control signal for the valve V_1, ($V_1 = 1$ or $V_1 = 0$). The FSA will change state if an internal transition is detected, ($\theta_1 = 1$ or $\theta_1 = 0$).

The *Fill Basin* function is refined by the FSA dis-

played in Fig. 9. The input for this FSA is the amount of control that is being applied to the system by valves V_1 and V_2. The output is how many truck loads of mixture the basin contains.

The function block *Mixture Volume in Truck* is refined by an FSA with three states. The input to this FSA is the amount of control applied by valve V_3, the system output is the state of the truck, whether the truck is *FILLING*, *FULL* or *OVERFLOWING*.

The control equation $\dot{x} = Ax + Bu$ is used to model the continuous state variables in the system. Where x is the subsystem state, u is the control from the valves, and B is the amount of control being applied.

6.2 Model Execution

A coordinator is used to create and execute the multimodel system. After each level of the model is created, the levels are connected to their refining models, then each level is initialized. Our initial conditions for the truck depot example are the basin is empty, the valves are closed, and no trucks are waiting to be serviced. The coordinator's task during execution is to dequeue events from the FEL and direct the event to the model specified.

7 Conclusions

Through our truck depot example, we demonstrated how to integrate simulation and planning tasks under the object-oriented multimodel framework. The designing of the model for our system is performed through the 3 phases 1) concept model, 2) class model and 3) instance model with the relationships specified as discussed in section 1. As shown, multimodeling cannot only be used to integrate different type models in a hierarchy but also used to integrate model types coming from completely different background or disciplines.

For future work, we would first like to extend our truck depot example to include more dynamic properties such as varying tanker truck capacity and allowing the planner to reorder the trucks after they arrive. We would also like to experiment with different intelligent objects, such as a mobile robot and adding more constraints to the problem such as asynchronous control of the valves. Using adaptive control (e.g. adaptive fuzzy systems) will be another extension. For multimodeling in general, we will add a graphical interface for creating the models and a distributed persistent object database to store all of the objects which have been created. Then a user could load any object from the Internet.

Acknowledgments

We would like to thank the Institute for Simulation and Training (IST) at the University of Central Florida for partial funding of this research in connection with the "Mission Planning" sub-contract #307043.

References

[1] G. Booch. *Object Oriented Design.* Benjamin Cummings, 1991.

[2] R. A. Brooks. A robot layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2:14 – 23, 1986.

[3] T. L. Dean and M. P. Wellman. *Planning and Control.* Morgan Kaufmann, 1991.

[4] P. A. Fishwick. Heterogeneous Decomposition and Coupling for Combined Modeling. In *1991 Winter Simulation Conference*, pages 1199 – 1208, Phoenix, AZ, December 1991.

[5] P. A. Fishwick. An Integrated Approach to System Modelling using a Synthesis of Artificial Intelligence, Software Engineering and Simulation Methodologies. *ACM Transactions on Modeling and Computer Simulation*, 2(4):307 – 330, 1992.

[6] P. A. Fishwick. A Simulation Environment for Multimodeling. (accepted for publication), 1993.

[7] P. A. Fishwick. *Computer Simulation Model Design & Execution.* Prentice Hall, 1993. (to be published as a textbook).

[8] P. A. Fishwick and B. P. Zeigler. A Multimodel Methodology for Qualitative Model Engineering. *ACM Transactions on Modeling and Computer Simulation*, 1(2):52 – 81, 1992.

[9] I. Futo and T. Gergely. *Artificial Intelligence in Simulation.* Ellis Horwood Limited/John Wiley and Sons, 1990.

[10] D. Harel. On Visual Formalisms. *Communications of the ACM*, 31(5):514 – 530, May 1988.

[11] D. Harel. Biting the Silver Bullet: Toward a Brighter Future for System Development. *IEEE Computer*, 25(1):8 – 20, January 1992.

[12] L. P. Kaelbling. An architecture for intelligent reactive systems. In *Reasoning About Actions and Plans*, pages 395 – 410. Morgan Kaufmann, Los Altos, CA, 1987.

[13] V. T. Miller. *Heterogeneous Hierarchical Modelling for Knowledge-Based Autonomous Systems.* PhD thesis, University of Florida, 1993.

[14] N. R. Nielsen. Applications of AI Techniques to Simulation. In P. Fishwick and R. Modjeski, editors, *Knowledge Based Simulation: Methodology and Application*, pages 1 – 19. Springer Verlag, 1991.

[15] R. M. O'Keefe. The Role of Artificial Intelligence in Discrete Event Simulation. In L. E. Widman, K. A. Loparo, and N. R. Nielsen, editors, *Artificial Intelligence, Simulation & Modeling*, pages 359 – 379. John Wiley and Sons, 1989.

[16] H. Praehofer. *Theoretic Foundations for Combined Discrete Continuous System Simulation.* PhD thesis, University Linz, Austria, 1991.

[17] H. Praehofer, G. Auernig, and Reisinger G. An Environment for DEVS-Based Modeling in Common Lisp/CLOSS. (accepted for publication), 1993.

[18] J. Rumbaugh, M. Blaha, W. Premerlani, E. Frederick, and W Lorenson. *Object-Oriented Modeling and Design.* Prentice Hall, 1991.

[19] L. A. Zadeh. Fuzzy Logic. *IEEE Computer*, pages 83 – 93, April 1988.

[20] B. P. Zeigler. *Object Oriented Simulation with Hierarchical, Modular Models: Intelligent Agents and Endomorphic Systems.* Academic Press, 1990.