EXPERIMENTAL RESULTS OF TIMED CELL-DEVS QUANTIZATION

Gabriel A. Wainer
Departamento de Computación
Facultad de Ciencias Exactas y Naturales
Universidad de Buenos Aires
Planta Baja. Pabellón I. Ciudad Universitaria.
(1428) Buenos Aires. Argentina
gabrielw@dc.uba.ar

Bernard P. Zeigler
Arizona Center for Integrated M&S.
Electrical and Computer Engineering Dept.
University of Arizona,
Tucson, AZ 85715.

zeigler@ece.arizona.edu

Keywords: Simulation methods: Discrete-event simulation Modeling methodology: DEVS models, Cell-DEVS models, quantization.

Abstract

An experimental analysis of quantized Cell-DEVS models is presented. The experiments show that execution times can be reduced according with bx^{-a} when quantized cell spaces are used. The error introduced has linear growth for small quanta. The experimental studies suggest how to define dynamic strategies to improve the execution times in timed Cell-DEVS. Quantization ideas are easy to apply, and the concept is generic to be used in any simulation environment.

INTRODUCTION

In (Wainer 1998) the Timed Cell-DEVS formalism was presented, as a combination of the DEVS (Zeigler 1976) and Cellular Automata (Wolfram 1986) paradigms with timing delays (Giambiasi and Miara 1976). The idea is to describe cell spaces as a DEVS model that can be delayed using several constructions. The DEVS paradigm was taken as base, due to it is a formal approach to specify discrete events systems using a modular description.

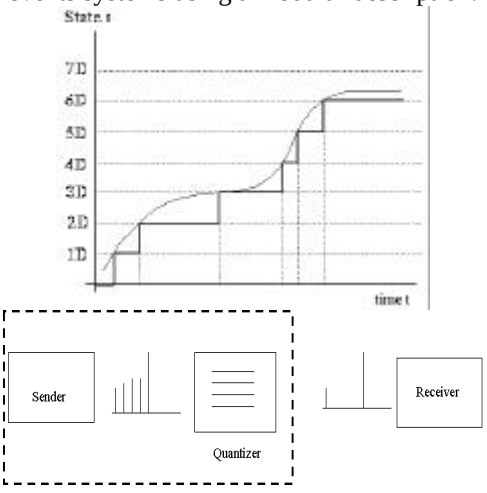


Figure 1. Quantization (Zeigler et al. 1999)

This work was partially supported by ANPCYT Project 11-04460 and UBACYT Project JW10.

Recently, a theory of quantized models was developed (Zeigler 1998, Zeigler et al. 1998). The theory has been verified when applied to predictive quantization of arbitrary ordinary differential equation models. A curve is represented by the crossings of an equal spaced set of boundaries, separated by a *quantum* size. A *quantizer* checks for boundary crossings whenever a change in a model takes place. Only when such a crossing occurs, a new value is sent to the receiver. This operation reduces substantially the frequency of message updates, while potentially incurring into error.

The cost/benefit analysis between reduced traffic and increased error was discussed in (Zeigler et al. 1999). The goal of this work is to show the applicability of the approach when used in timed Cell-DEVS.

The following sections will be devoted to present several empirical results obtained using the quantization theory in timed Cell-DEVS. First, a review on DEVS and Cell-DEVS is presented. After this, the main results related with performance improvements are shown. Finally, the models' error behavior is characterized.

BACKGROUND: DEVS FORMALISM

A DEVS model is seen as composed atomic submodels than can be combined into coupled models. A DEVS atomic model is described as:

$$M = < I, X, S, Y, \delta_{int}, \delta_{ext}, \lambda, D >$$

Here, ***I*** is the model's interface, ***X*** is the input events set, **S** is the state set, and ***Y*** is the output events set. There are also several functions: δ_{int} manages internal transitions, δ_{ext} external transitions, λ the outputs, and **D** the elapsed time.

A DEVS coupled model is defined as:

$$CM = < I, X, Y, D, \{M_i\}, \{I_i\}, \{Z_{ij}\} >$$

Here, ***I*** is the model's interface, ***X*** is the set of input events, and ***Y*** is the set of output events. **D** is an index of components, and for each i ∈ D, $\mathbf{M_i}$ is a basic DEVS model, where $M_i = < I_i, X_i, S_i, Y_i, \delta_{inti}, \delta_{exti}, ta_i >$. $\mathbf{I_i}$ is the set of influencees of model i. For each j ∈ I_i, $\mathbf{Z_{ij}}$ is the i to j translation function.

Timed Cell-DEVS allows us to define cellular discrete events models. Each cell is defined as an atomic DEVS model, and a procedure to couple cells is depicted. Transport and inertial delays allow to define timing behavior. A *transport* delay allows us to model a variable commuting time for each cell with anticipatory semantics (every schedled event will be executed). Using *inertial* delays, the semantics is preemptive: some scheduled events are not executed due to a small interval between two input events. This kind of delay allows us to analyze frequency limit response of systems.

Recalling the definitions from (Wainer 1998), Cell-DEVS atomic models can be formally specified as:

$$TDC = < X, Y, I, S, \theta, N, delay, d, \delta_{int}, \delta_{ext}, \tau, \lambda, D >$$

In this case, ***X*** is the set of external input events, ***Y*** is the set of external output events, and ***I*** is the model's modular interface. **S** is the set of sequential states for the cell, **θ** is the cell state definition, and **N** is the set of input events. **Delay** defines the kind of delay for the cell, and **d** its duration. Finally, there are several functions: δ_{int} for internal transitions, δ_{ext} for external transitions, **τ** for local computations, **λ** for outputs and **D** for the state's duration function.

A Cell-DEVS coupled model is defined by:

$$GCC = < Xlist, Ylist, I, X, Y, n, \{t_1,...,t_n\}, N, C, B, Z >$$

Here, **Ylist** is the output coupling list, **Xlist** is the output coupling list and ***I*** represents the definition of the model's interface. ***X*** is the set of external input events and ***Y*** is the set of external output events. The **n** value defines the dimension of the cell space, $\{\mathbf{t_1,...,t_n}\}$ is the number of cells in each dimension and **N** is the neighborhood set. **C** is the cell space, **B** is the set of border cells, and **Z** the translation function.

QUANTIZED CELL-DEVS EXECUTION

Several experimental tests were done in order to analyze the behavior of quantized Cell-DEVS models. The quantized version includes a quantum to define the state change for each cell in the model. Two classes of behavior were analyzed: the execution time, and the number of messages involved during execution.

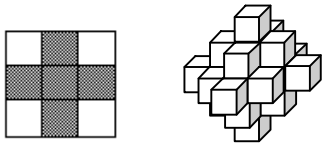


Figure 2. Cell's neighborhood. (a) Von-Neumann Neighborhood. (b) 3-dimensional heat model.

The following models were executed using a quantized version of the N-CD++ tool (Rodríguez and Wainer 1999):

a) Heat diffusion: a two-dimensional model (10x10 cells) executed during 1 minute simulated time. One cell is "hot" and the rest remain without initial heating. The neighborhood shape can be seen in Figure 2a.

b) Heat model with 87% of active cells.

c) Three-dimensional extension of the previous model. The neighborhood shape is shown in Figure 2b.

d) Three-dimensional modification of the Life game (10x10x10 cells). The neighborhood includes the cell (0,-8) to the South in the XY plane. The cell's value changes according with the neighborhood, averaging them in some cases, multiplying them in others. The goal is to introduce complex neighborhood behavior and real numbers for the state variables. It was executed for 20 seconds simulated time.

e) Four dimensional extension of the previous model. It was executed for 20 seconds simulated time.

f) Dynamic heat seeker: a three dimensional model consisting of two adjacent planes. One of the surfaces executes a heat diffusion model. The other one includes a set of heat-seeking devices that follows the heat cells towards a local maximum. It was executed for 3 seconds simulated time.

The number of messages involved in the execution for Cell-DEVS spaces can be expressed as follows:

$$m_i = \sum_{j=1}^{i} n_j \cdot \mu \qquad (1)$$

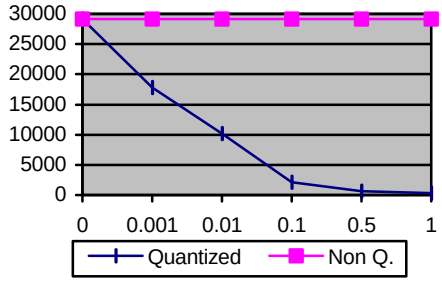
m_i: number of messages distributed up to the i-eth simulation step;

n_j: number of active cells in the *j*-eth simulation step.

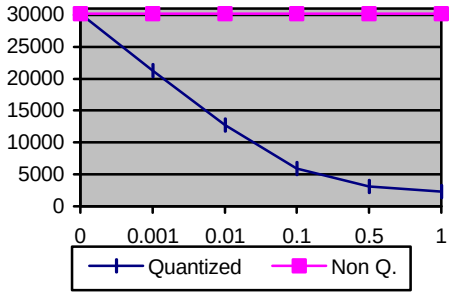
μ: neighborhood size.

The results presented in the following figure show a reduction according to bx^{-a} in the number of messages involved. Analyzing equation (1), it can be seen that n_j has been reduced in each time step. Also, the use of a quantized

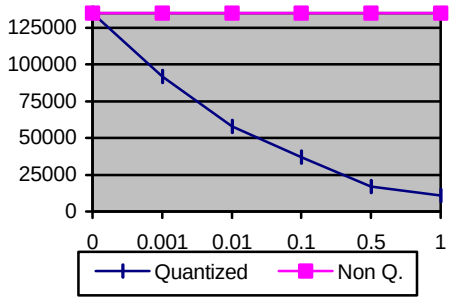
version provides fewer steps to be executed, reducing the *i* value in the equation.



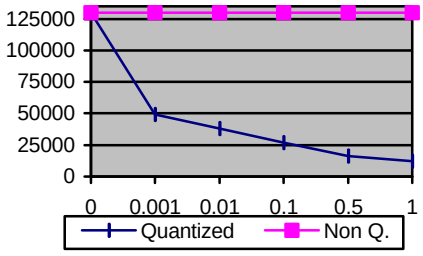
(a) $f(x) = 441x^{-0.58}$



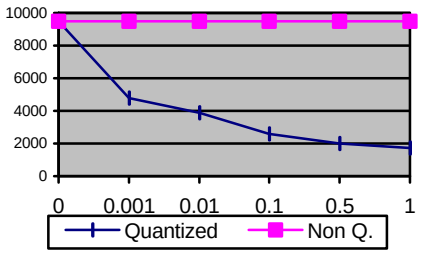
(b) $f(x) = 2547x^{-0.32}$



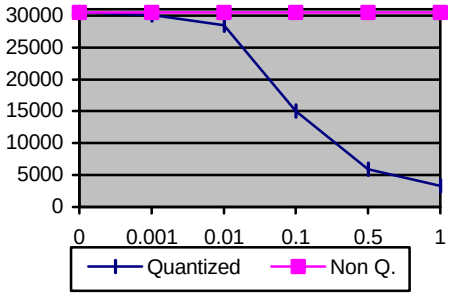
(c) $f(x) = 13752\ x^{-0.3}$



(d) $f(x) = 14013\ x^{-0.2}$



(e) $f(x) = 1860x^{-0.15}$



(f) $f(x) = 4790\ x^{-0.32}$

Figure 3. Number of messages involved.

The curves belong to the class of curves $f(x) = bx^{-a}$ with $x \in (0,1]$. Approximations of *a* and *b* values were found using the minimum square method, allowing to find a description for each curve, that are presented in the figure. These results approximate the theoretical optimum results presented in (Zeigler et al. 1998).

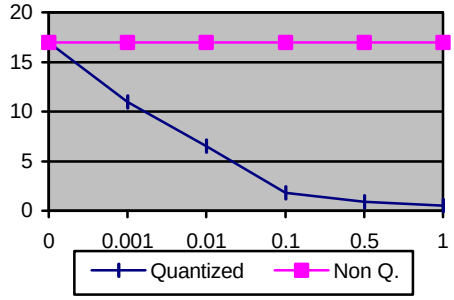
Message reductions in model (b) are slightly less than the (a) case, because the number of active initial cells is higher. Therefore, in the first simulation steps, n_j is greater than the previous case. The accumulation of these values makes that reduction slightly slower than in model (a).

The case is different for model (c) though the curve shapes are similar. Here, we can see that the number of messages has increased significantly, reflecting the change of the neighborhood and cell space sizes. The proportion of active cells is significantly reduced (4 initial hot cells in a total of 1000). Therefore, the value μ has the main weight in equation (1), producing a slower reduction for high quanta. Similar results were achieved for model (e). In this case, the number of active cells is proportionally small in the four dimensional model.

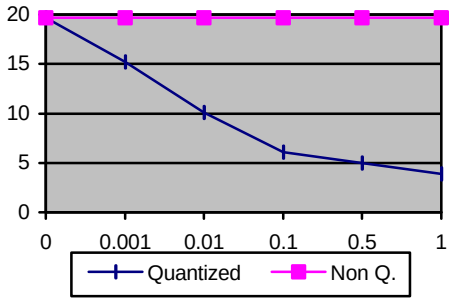
The model (d) uses about 60% of active cells. The neighborhood size has been reduced to 7 cells. The number of messages is similar to the previous case, though the simulated time was reduced to 33%. Therefore, n_j increases while μ was reduced proportionally.

In model (f), the number of active cells between 0 and 0.001 changed around 1% and 5%. This happened because the initial values for temperatures had a precision of 0.1 degrees. Therefore, the average temperature differs in 0.01 or 0.001 sizes only after long execution time, while the simulation run for 3 seconds (the total seeking time).

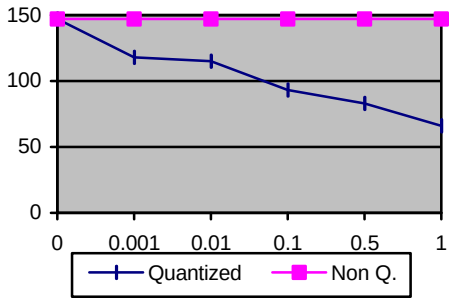
Now we will analyze the execution times for the same models studied previously. The results obtained are presented follows:



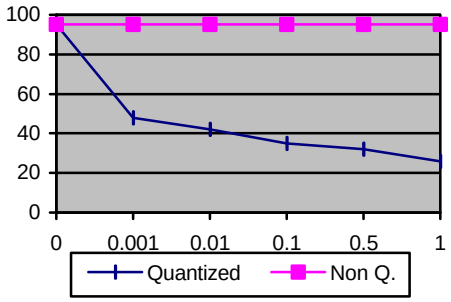
(a) $f(x) = 0.61\ x^{-0.45}$



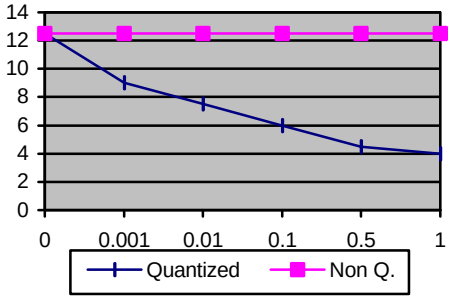
(b) $f(x) = 4.1\ x^{-0.19}$



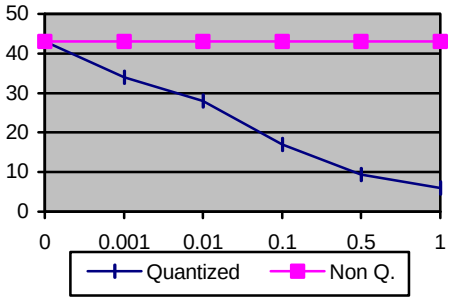
(c) $f(x) = 73\ x^{-0.08}$



(d) $f(x) = 28.3\ x^{-0.08}$



(e) $f(x) = 4.2\ x^{-0.12}$



(f) $f(x) = 7.67x^{-0.24}$

Figure 4. Execution time for the examples.

In this case, the total execution time can be expressed as:

$$t_i = \sum_{j=1}^{i} [\ (n_j \,.\, \mu \,.\, x_j) + (n_j \,.\, \tau_j)\ ] \qquad (2)$$

t_i: total execution time up to the i-th simulation step;
n_j: number of active cells in the *j*-th simulation step;
x_j: transmission time for each message;
μ: neighborhood size; and
τ_j: execution time for the local computing function.

The results obtained in models (a) and (b) are proportional to those obtained analyzing the number of messages involved. In this case, the execution and transmission times for each cell are equivalent. The results obtained with larger quantum have increased proportionally to the number of messages involved. This is because the

first step of the simulation (where the quantum size has no influence) introduces a minimum execution time.

Model (c) obtained a smaller run time reduction than the number of messages involved, due to the size of the neighborhood. Initially, there are 4 "hot" cells with 27 neighbors each. Therefore, the transmission time is at least of $112*x_j$. A similar phenomenon occurs in case (e). In this case the difference is slightly lower than in the (c) case, because here we have only 10 neighbors. Preparation time for messages in 3 and 4-dimensional spaces produces higher amounts of overhead. This can be seen analyzing the (e) model. It has the same neighborhood and computing function of (d), but the messages has to be sent to 4-dimensional neighbors.

An interesting behavior can be seen in model (f). Even though the number of messages was not reduced for small quanta, the execution speed improved. The model is divided in two planes, each executing different computing functions. One of them averages the values of the neighbors, while the other compares the present values. The computation of the average is more time consuming than the comparisons of the seeker plane. Analyzing the log files, the cells of the heat surface becomes inactive faster due to the quantized version. Even the use of a small quantum does not reduce the number of messages, improvement in the execution speed is achieved, due that this is the more time consuming part of the model.

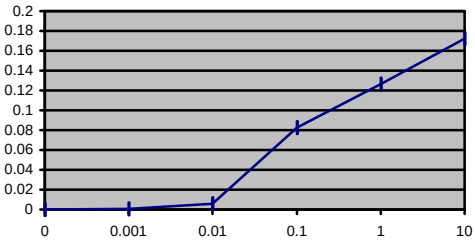
MODEL'S ERROR BEHAVIOR

The error behavior of these models can be expressed as:

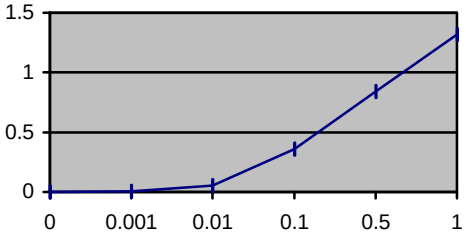
$$e(C_c, i) = \sum_{j=1}^{i} |\tau_{cj}(N_c) - [\tau_{cj}(N_c)]_q| \qquad (3)$$

Here, $e(C_c, i)$ is the accumulated error up to the i-th simulation step in cell C_c (c is an n-dimensional index of the cell). N_c are the inputs of the cell c, τ_{cj} is the execution result of j-th step of the local computing function of cell c, and $[\ ]_q$ represents the quantized value of the last change.

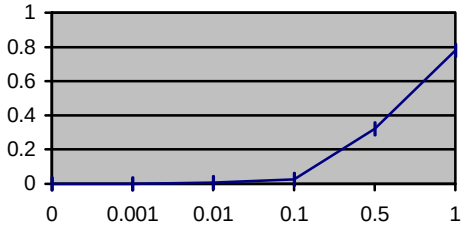
The error obtained is a function of the local computing function, the number of simulation steps and the quantum. The future input values for a cell are dependent of the present results for the cell. This can lead to a nonlinear behavior of the error, depending on the cell's interconnection. In any case it can be seen that the higher the quantum, the worse the error. The use of a higher quantum reduces the number of steps, but each of them will have higher error. The experimental results validate this behavior.



(a) $f(x) = 0.1\ x^{0.63}$



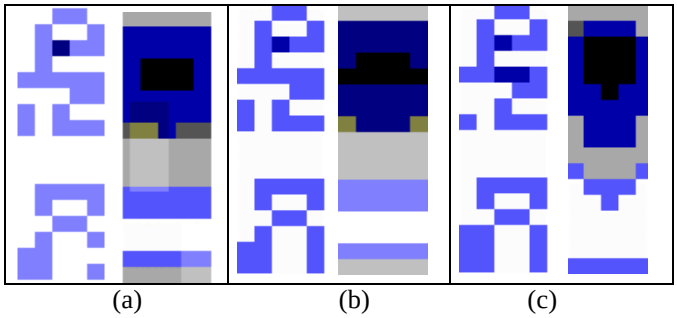
(c) $f(x) = 1.58\ x^{0.77}$



(d) $f(x) = 0.57\ x^{1.02}$

Figure 5. Accumulated error behavior.

It can be seen that the error grows as $f(x) = ax^b$. This error can be linear when there is no influence between cells. We can see that, in the (a) case, the error hardly increases while the messages go down by approximately 1/10. Nevertheless, the error can lead to undesired behavior for the execution models, as shown in the following figure.

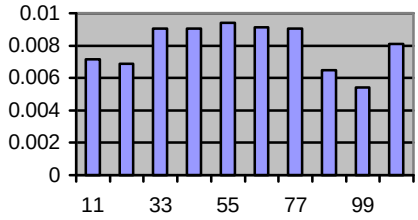


(a) (b) (c)

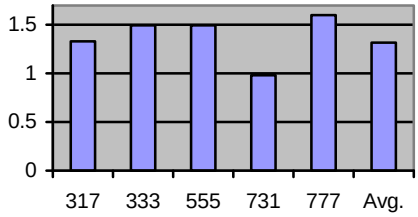
Figure 6. Error behavior for the heat seekers. (a) non-quantized and q=0.001; (b) q=0.1; (c) q=1.

The figure shows the seekers in the left, and the surface in the right for a given simulated time (using different quantum in each case). We can see that the use of a high quantum produces much higher errors, producing a final result completely different from the desired. The use of a smaller quantum also introduces errors, but most seekers can reach the local maximum. Therefore, several orders of improvement in execution times can be achieved using small quanta, preserving the model's behavior. In this case, a speedup of almost 40% was achieved with almost no error.

Another set of tests was devoted to analyze the error behavior depending on the individual cells. The results can be seen in the following figure:



(a)



(b)

Figure 7. Error behavior for individual cells. (a) Model (a); (b) Model (c).

The error is reduced when the cells analyzed are far from the more active ones. This allowed us to define two dynamical quantum adjustment strategies:

a) Reduce the quantum of the most inactive cells. The goal of this policy is to improve the precision of the inactive cells. An active cell can appear as quiescent due to the use of a quantized version. Therefore, if the quantum is reduced, the error introduced can be avoided. Besides, the quantum is increased for the most active cells, thus improving execution times.

b) Increase the quantum of the most inactive cells. This would lead to faster inactive behavior, eliminating these cells from the computations. Besides, the most active cells will have higher quanta, thus reducing their error.

CONCLUSION

Several conclusions can be drawn of the execution of the present examples and of those reported in (Zeigler 1998, Zeigler et al. 1998 and Zeigler et al. 1999):

- The number of messages involved in the execution diminishes according to bx^{-a}. This result is especially useful in distributed environments, where the communication costs can produce degradation in the execution times.
- Due to the transient characteristics of the examples in this paper, the main reduction is due to the number of active cells involved. The use of a quantum deactivates the cells in fewer simulation steps. The shape and size of the neighborhood and the dimension of the cell space are also influences. Recent results from other studies indicate that message reduction can be independent of active cell numbers.
- Quantum size increase can reduce the number of simulation steps.
- Equation (2) introduces a factor for transmission time for messages. This value was small in the presented examples, due to the use of a centralized simulation approach. A distributed version would increase this value, making the quantized approach even more useful.

The quantization process can be used in experimental phases of complex systems studied through simulation. In these cases it takes a long time to find a specification's errors, leading to a time consuming procedure. Quantized versions can be used to obtain fast results in the early testing phases.

Finally, the quantization approach is relatively easy to apply and requires minimal restructuring of the state computation processes. However, it can incur loss of accuracy due to the receiver's diminished state updates and this may propagate in a global error due to feedback between sender and receiver. The concept is generic and can be employed in any distributed simulation environment.

REFERENCES

Giambiasi, N. and Miara, A. "SILOG: A practical tool for digital logic circuit simulation. In Proceedings of the 16th. D.A.C., San Diego, U.S.A. 1976.

Rodríguez, D. and Wainer, G. "New Extensions to the CD++ tool". In Proceedings of SCS Summer Multiconference on Computer Simulation, Chicago, U.S.A. 1999.

Wainer, G. "Discrete-events cellular models with explicit delays". Ph.D. Thesis, Université d'Aix-Marseille III. 1998.

Wolfram, S. "Theory and applications of cellular automata". Vol. 1, Advances Series on Complex Systems. World Scientific, Singapore, 1986.

Zeigler, B. Theory of modeling and simulation. Wiley, 1976.

Zeigler, B. DEVS Theory of Quantization. DARPA Contract N6133997K-0007: ECE Dept., UA, Tucson, AZ. 1998

Zeigler, B.; Cho, H. ; Lee, J. and Sarjoughian, H. The DEVS/HLA Distributed Simulation Environment and its Support for Predictive Filtering. DARPA Contract N6133997K-0007: ECE Dept., UA, Tucson, AZ. 1998.

Zeigler, B.; Ball, G.; Cho, H. Lee, J. and Sarjoughian, H. "Bandwidth Utilization/Fidelity Tradeoffs in Predictive Filtering". In SISO SIW '99. Orlando, Florida. March 1999.