*Proceedings of the 1992 Winter Simulation Conference*
*ed. J. J. Swain, D. Goldsman, R. C. Crain, and J. R. Wilson*

# A SOFTWARE MECHANISM TO ENHANCE SIMULATION MODEL VALIDITY

Gaynor Legge and Dana L. Wyatt

Dept. of Computer Science
University of North Texas
Denton, Texas 76203

## ABSTRACT

This paper introduces a software mechanism which can be incorporated into standard simulators, facilitating the development of valid models. This mechanism allows modeling of situations which might otherwise be difficult to simulate accurately. The mechanism implements timelines which are used to characterize problematic situations. The object oriented simulator and the mechanism are described and then applied to two example models. Each model is simulated both with and without the mechanism in order to verify that the mechanism operates correctly. The ease with which each version of each model is implemented provides a basis for comparing the standard and enhanced versions of models.

## 1 INTRODUCTION

There are some real world situations that are difficult or impractical to model closely using established discrete event simulation methods and languages. This is because most computer simulation languages limit access to the event list and individual transactions, thereby limiting the range of behaviors which can be simulated. In these cases the real world situation is modified or generalized to allow for computer modeling language capabilities. However, modifications and generalizations of real world situations can adversely affect the validity of a model.

### 1.1 Problematic Real World Situations

This paper examines two situations which pose problems for modelers. The first of these is the occurrence of an independent event which affects one or more of the objects which move through a system. The second concerns the influence which expected future events might have on current events.

An example of the first situation is one in which an airplane has started its landing descent but is waived away because of the arrival of another plane requiring an emergency landing. In a discrete event simulation, the landing event of the first plane would already be on the event list, having been scheduled at the time the plane started its approach. It would be necessary to remove this event, either at the time the emergency occurs or when the clock is moved forward to the time when the landing was supposed to have taken place. In order to remove the landing event at the time of the emergency, (1) the simulation must 'know' that there is a plane on approach to land, and (2) the event list must be searched to find and remove the already scheduled landing event. To cancel the landing at the time it was to have taken place, the simulation must have a way to 'remember' the intervening emergency and temporally connect it to the aborted landing. It would also be necessary to schedule a new landing approach for the first plane.

The influence of future events on current events is even more difficult to model well. El Sheikh et al., in a somewhat complex harbor simulation, found that expected future events affected their model (El Sheikh et al. 1987). In the real world situation, a harbor master was scheduling dock space depending on what kind of ship was expected to arrive. For example, if a ship of type A, needing a dock of type X, was due to arrive shortly, the harbor master would not assign this dock to another waiting ship of type B, even though the dock was idle. In their simulation model, however, the dock of type X was assigned to ship B. Then, when ship A arrived, ship B was preempted and returned to the queue of ships waiting for a dock. However, the preemption method creates some complications for the modeler. How should the waiting time for ship B be adjusted? What if, while ship B was at dock X, another dock became free but was assigned to ship C? Should ship B then preempt ship C? Regardless of how these questions are resolved, there will be some distortion of the simulation results. Whether the distortion will be significant or not depends on the model and the model parameters.

### 1.2 Project Overview

The identification of real world situations which are difficult to simulate evokes speculation concerning possible improvements to current discrete event simulation methods. The basic premise of this paper is that real world situations can be modeled more accurately by adding a software mechanism which is able to recognize specific problem situations. The mechanism is called DELIM, Discrete Event List Inference Mechanism. It is applicable to situations which involve the intervention of independent events or the effect of future events on current events. DELIM provides substance to the concept of timelines by applying them to actual situations (Legge and Wyatt 1991).

## 2 TIMELINES

The event list of a simulation is a form of temporal representation in which the temporal relationships of before and after are implicitly defined. Because these relationships are inherent in an event list, a more abstract form of the event list called a timeline can be used to delineate temporal relationships between events without reference to specific times.

A timeline is a sequence of events ordered by their relative time of occurrence. The events on a timeline are ordered from left to right, and the only temporal relationships represented are Event A < Event B (meaning Event A precedes Event B) and Event B > Event A (Event B takes place after Event A.) The timeline does not show the actual time an event takes place, only the relevant order of the events on the timeline. Events are represented as capital letters (e.g. A, B, C). Timelines are used to compare one sequence of events to another. The two timelines in figure 1 are sequences of events which can be difficult to model using standard methodology.

```
Timeline    |--------A-----------B----------
            |--------A---C--------?----------

                 a. Intervention

Timeline    |--------A--------B--------------
            |--------A--------?--------C-----

                 b. Anticipation
```

Figure 1: Two types of time sequences

The concept of specifying a timeline of events and then matching it to a real event queue can be applied to simplify the development of models which include these timelines. The addition of a mechanism which does this requires minimal adjustment to a standard simulator. More importantly, it allows the simulation of the intervention and anticipation timelines in a consistent and more uniform way than could be achieved by modeling each situation as a specific case. Both the anticipation and intervention situations can be represented using the same rule format, thereby reducing the amount of specialized program code and simplifying the modeling task (Legge 1992).

## 3 THE SIMULATION SYSTEM

DELIM and the basic simulator to which it is connected are programmed in the object oriented language C++ (Stroustrup 1986). The simulator consists of seven objects or base classes. Some of these objects may be tailored to include characteristics unique to a given model. There are three types of model objects: transactions, servers and queues. A server performs a task or service of some kind. A transaction moves around the system visiting one or more servers; it is the object on which a service or task is performed. A queue is a waiting line for transactions which must wait for service at a server.

There are four types of system objects: the simulator object, a clock, an event list and events. The clock maintains the system time. Events are used to mark the times of arrivals and departures of transactions during the simulation. The event list is a list of events ordered by their time of occurrence. The simulator is the object which manages the event list, processes each event, advances the clock as each event occurs, and starts and stops the simulation.

DELIM consists of five classes of objects. The subframe, substitution and matchframe objects are all constituents of the rule object. The rule objects are contained in a mechanism object, i.e. DELIM.

### 3.1 Simulation Function Overview

The functioning of the simulation system without DELIM is straightforward. A model specific simulator is derived from the simulator base class. It includes the servers needed for the simulation. The servers may themselves be derived classes. Any necessary queues are also included, either in the servers or the simulator. The user sets the end time for the simulation, and the simulation is initialized by invoking the appropriate method of the simulator.

If the model has transient objects, the arrival of a new transaction causes a new arrival event to be created and inserted into the event list. The arriving transaction is sent to the first server where either (1) the service time is determined and a departure event is returned to the simulator and inserted into the event list or, (2) the transaction is put into the queue to wait until the server is free. When a transaction leaves a server, it either leaves the system or an arrival event at the next server is created and placed in the event list. The server either becomes idle or a waiting transaction is dequeued, its service time computed and a new departure event is returned to the simulator which inserts it into the event list.

### 3.2 DELIM Function Overview

The main function of DELIM is to match a representation of a timeline, or rule, to the actual events on an event list in a model simulation. The key element of DELIM is the rule, which is a list of abstract events which may or may not be matched to actual events on an event list during a simulation.

DELIM operates in a manner similar to the inference engine of an expert system (Charniak and McDermott 1984). Using a representation of a timeline as the condition part of a rule, DELIM takes the current event of an event list and binds it to the 'current' event of the rule. It then searches the future and/or past of the event list, as appropriate, to match events in the rule to events on the list. If the events specified in the rule are matched on the event list, the rule is fired, i.e. some instruction is returned to the simulator which executes events and manages the event list. The length of a search along the event list is limited, when necessary, by parameters associated with the current rule.

There is additional leeway allowed in the portion of an event which is matched. Minimally, the type of event is matched. Depending on the construction of the rule, however, it is possible to match both the event type and a specific transaction, or the event type and the transaction type, or the event type and a specific transaction of a designated type.

The links to the simulation system consist of a reference to the event list and the return to the simulator of an indication of either a successful match to a specific rule or failure (see figure 2). Unlike an expert system, however, DELIM does not alter the knowledge base, in this case the event list, nor can DELIM actually be said to perform inference, since only one rule is ever fired when DELIM is invoked. Therefore, although the method of its operation has been adapted from that of expert systems, a more appropriate description of DELIM's function would be a timeline recognizer.

A significant characteristic of DELIM is its generality. Although its purpose is to recognize sequences of events represented by timelines, it is sufficiently general to recognize any variations of timelines also. For example, the letter A on a timeline might represent two or three actual events. Since there are no restrictions to the number past or future abstract events which may be added to a rule, this sort of variation is easily handled. Furthermore, a rule may have both past and future abstract events in addition to the current event, a configuration outside the purview of the present set of timelines.

## 4 APPLICATION TO MODELS

In order to evaluate DELIM, it must be applied in simulation models. To that end, one model corresponding to each of the timelines in figure 1 has been developed. The models were selected from modeling literature to meet several criteria. First, corresponding to each of the timelines in figure 1 has each model demonstrates a single timeline in order to allow the independent examination of DELIM's efficacy in each situation. Second, each model reflects plausible situations in the real world. Since the primary reason for simulation is to model real world situations, it would be ineffectual to demonstrate DELIM on a timeline for which no realistic scenario exists. Finally, each model is of sufficient simplicity to clearly demonstrate the timeline it models without introducing extraneous events which could cause the effect of DELIM to become ambiguous.

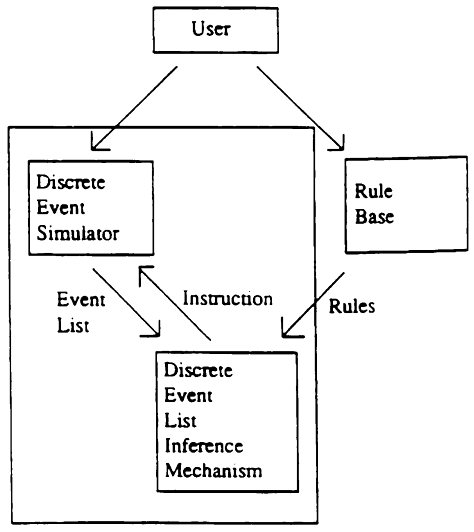

Figure 2: The system architecture

Two simulation models in a discrete event scheduling conceptual framework are implemented both with and without DELIM for purposes of comparison.

The technique employed is to start with the standard simulation system, that is, the seven base classes of the simulation system without DELIM, and derive the necessary objects to implement each model using standard discrete event simulation methodology. After implementing each model as accurately as possible using a standard technique, the models are implemented using the extended system incorporating DELIM. Thus differences between the two implementations are examined with the expectation that they provide insight into the evaluation of DELIM.

## 4.1 Intervention Model

The intervention model is comprised of a tanker fleet which travels between two ports. It is adapted from a GASP textbook model (Pritsker 1974). The tanker fleet has:

* 15 tankers
* a tanker is loaded with oil in Valdez and unloaded in Seattle
* load time is exponential with a mean of amount_loaded / average_loading_rate
* unload time is 200 tb/day
* travel from Valdez to Seattle takes 3.5-6.5 days
* travel from Seattle to Valdez takes 3-5 days
* 15 docks in Valdez, 1 dock in Seattle

However, storms occur which can delay ships in transit for as much as 2 days. Storm occurrences are normally distributed with a mean of 30 days and a standard deviation of 15 days. The tanker fleet is simulated for 365 days.

### 4.1.1 Comparison to the Intervention Timeline

Figure 3 matches the events on the intervention timeline to the sequence of events in the intervention model.

```
Timeline  |-------A--------------------B-----------
            leave port A         arrive port B

          |-------A---------C--------------?-------
            leave port A  storm     delayed arrival
```

Figure 3: The intervention timeline and model sequence

### 4.1.2 Standard Implementation

The simulator for the intervention model contains a dock server for Alaska, a dock for Seattle, and a queue. Tanker objects are derived from the transaction class. In the standard implementation, a storm notice is maintained to record the times of storms as they occur. A new method is added to compare the time of a tanker's departure from a port with the times of the most recent storm. The simulator creates all fifteen tankers at the beginning of the simulation. Arrival events at the Alaska dock for each tanker are generated to take place at half day intervals and all of them are inserted into the event list. The first storm event is also created and scheduled.

The simulation proceeds as follows for each type of event:

```
Arrival:
  tanker arrives at port
  if tanker_departure_time < storm_notice_time then
    calculate delayed arrival_time
    reset tanker_departure_time to current time
    schedule new arrival event
  else
    tanker goes to dock
    if dock is not busy then
      schedule new departure event
    else
      tanker is put on the queue

Departure:
  tanker leaves port
  release the dock
  schedule new arrival event
  if tankers are waiting then
    remove a tanker from the queue
    schedule new departure event

Storm:
  storm occurs
  post time of storm in storm_notice variable
  schedule new storm event
```

### 4.1.3 DELIM Implementation

The same objects are used for the DELIM implementation as for the standard implementation, that is, two docks and a queue object located in the simulator. The simulator object no longer needs to have a storm notice or the method associated with it. The tanker departure time is changed to a flag variable to prevent a tanker from being repeatedly delayed by the same storm. The changes required to the event routines are minimal:

```
Arrival:
  tanker arrives at port
  if no delay_flag and
    DELIM returns a rule number then
      calculate delayed arrival_time
      set tanker delay_flag
    schedule new arrival event
  else
    tanker goes to dock
    clear delay_flag
    if dock is not busy then
      schedule new departure event
    else
      tanker is put on the queue
```

```
Storm:
  storm occurs
  schedule new storm event
```

The departure event sequence remains the same. The two rules needed for Model Two are displayed in figure 4. DELIM tries each of the rules in its rulebase in order and returns the number of first rule which succeeds. If some rules are more specific than others, they must be attempted first.

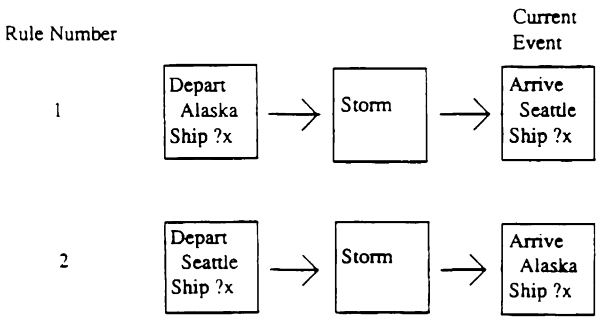

Figure 4: Intervention model rules

### 4.1.4 Intervention Model Results

The intervention model results for both the standard and DELIM simulator systems are shown in figure 5. The statistics reported at the end of each simulation run are identical, which is the anticipated and correct outcome.

## 4.2 Anticipation Model

The anticipation model is adapted from the same source as the intervention model (Pritsker 1974). A similar but more complex real world model involving the same timeline is found in the description of a harbor simulation developed to model an unnamed port (El Sheikh et al. 1987). In this model, the scenario is similar to the intervention model except that there are two types of tankers, regular and super-tankers. This model has:

* 12 regular tankers
* 3 super-tankers
* super-tankers carry a larger load

| Type of Statistic | Standard Model | DELIM Model |
|---|---|---|
| Average round trip time | 13.74340 | 13.74340 |
| Number of storms | 12 | 12 |
| Alaska dock utilization | 0.23492 | 0.23492 |
| Number of ships at port | 402 | 402 |
| Average loading time | 3.20232 | 3.20232 |
| Seattle dock utilization | 0.78377 | 0.78377 |
| Number of ships at port | 392 | 392 |
| Average unloading time | 0.73041 | 0.73041 |
| Average wait time | 0.91563 | 0.91563 |
| Maximum queue size | 5 | 5 |

Figure 5: Intervention model results

* there are 2 docks in Seattle, 1 regular and 1 special
* regular tankers may unload at either dock
* super-tankers must use the special dock
* super-tankers take priority over regular tankers

Super-tankers never have to wait for the unloading dock unless another super-tanker is using it. In addition, there are no storms; variations in travel time are due to weather conditions and other uncertainties associated with ocean travel. The model is simulated for 365 days.

### 4.2.1 Comparison to the Anticipation Timeline

The sequence of events which conforms to the anticipation timeline begins with the arrival of a regular tanker to the Seattle harbor. If the regular dock is busy and the super-tanker dock (super-dock) is free, the tanker may unload at the super-dock. However, in order to prevent an arriving super-tanker from waiting while a regular tanker unloads, the regular tanker should not unload at the super-dock if a super-tanker is expected within the time it would take the regular tanker to unload. Figure 6 shows this event sequence and its associated timeline.

```
Timeline  |-------A----------------B----------------
           tanker arrives     tanker
                              leaves dock S

          |-------A----------------------C---------
           tanker arrives    super-tanker arrives
```

Figure 6: The anticipation timeline and model sequence

### 4.2.2 Standard Implementation

The implementation of the anticipation model is the more complicated of the two models and requires changes to all of the basic objects of the simulator except the clock and event objects. The method for modeling the anticipation predicament with the standard simulation system is adopted from the method used by El Sheikh et al. in their harbor simulation model (El Sheikh et al. 1987):

```
Arrival of regular tanker:
  if regular dock not busy then
    schedule departure event from regular dock
  else
    if super dock not busy then
      schedule departure event from super-dock
    else
      put tanker on queue

Arrival of super-tanker:
  if super-dock not busy then
    schedule departure event from super-dock
  else
    try to preempt super-dock
      if preempt successful then
    restore load data for preempted tanker
    adjust some statistics
    remove tanker's scheduled departure event
    if regular dock not busy then
      schedule departure event for preempted tanker
    else
      put preempted tanker on the front of the queue
        schedule departure event for super-tanker
   else
     put super-tanker on queue
```

A tanker will never be preempted twice because after a preemption, the regular dock will be available before the super-dock. Although the dock utilization and count of tanker visits can be adjusted, the waiting time statistics for the tankers are distorted whenever a preemption occurs, as are the round trip times.

A new event list class must be derived from the eventlist class in order to add a method which can search for a specific event on the list and remove it. A specialized queue class is also necessary for this model because a method which puts a tanker onto the front of the queue is needed for the regular tanker queue.

A subtle problem arises in connection with the queue. It is possible, if a tanker is preempted, that it may enter the queue twice. The first time the queue would be entered normally at the back end when the tanker first arrives. Then, if the tanker is sent to the super-dock and gets preempted, it could be pushed back onto the front of the queue. In order to keep an accurate count of the number of ships passing through the queue, the tanker includes a flag, queueflag, to indicate whether or not it has already been in the queue. There are three methods associated with the flag which set the flag, access the flag and clear the flag.

This model has four classes of servers. The first is the Alaska dock, which is basically the same as the Alaska dock described in the intervention model. Next is the dock for regular tankers in Seattle. The dock for super-tankers is derived from the regular tanker dock class. It has a reference to a queue for super-tankers. Since this is the dock from which tankers are preempted, it has some extra variables and methods. The fourth class of server is the harbor, derived from class server. The harbor contains both the regular and super tanker docks in Seattle as well as the queues for each dock.

### 4.2.3 DELIM Implementation

Compared to the standard implementation just described, the DELIM implementation is relatively simple. No special event list or queue objects are required. The two different types of transaction objects, regular and super-tankers are retained because each type of tanker uses different methods to calculate the amount of oil which is loaded and unloaded.

There are four classes of server objects, as in the standard implementation, but no preemption methods or additional variables to allow preemption are necessary. The harbor contains two docks, two queues and a reference to DELIM. The super-dock needs no reference to its latest departure event or service time, although it does add a reference to DELIM. The regular tanker dock has a reference to a non-specialized queue object. The arrival actions are simpler:

```
Arrival of regular tanker:
  if regular dock not busy then
    schedule departure event from regular dock
  else
    if super-dock not busy & DELIM returns 0 then
      schedule departure event from super-dock
    else
      put tanker on queue

Arrival of super-tanker:
  if super-dock not busy then
    schedule departure event from super-dock
  else
    put super-tanker on queue
```

The two rules are necessary for this model are shown in figure 7. Each rule includes a time limit which is set to approximately the average amount of time it takes to unload a regular tanker, 0.8 time units. The time limit is determined by the modeler. In this case, if a tanker takes longer than average to unload, there is a possibility that a super-tanker could have to wait. Adjusting the time limit could reduce the probability that a super-tanker would wait at the expense of possibly making a regular tanker wait unnecessarily.

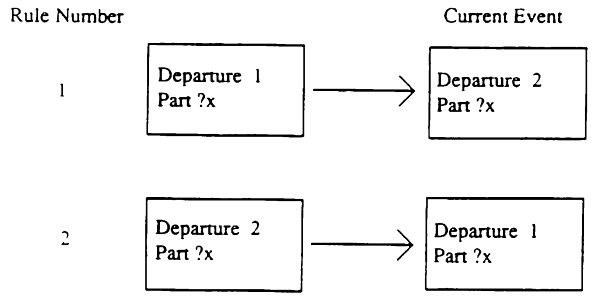

Figure 7: Anticipation model rules

### 4.2.4 Anticipation Model Results

Figure 8 shows that DELIM does affect the results of the simulation. This is the anticipated result, because there are no preemptions in the DELIM simulation. While the fact that the results of the two versions of this model are different is attributed to DELIM, the actual statistics produced by each simulation run are a result of the stochastic nature of the model. In other words, it cannot be said that the DELIM model produces more round trips because of the model behaves differently in the DELIM version. The increased number of round trips in the DELIM model is due to the lower average loading times for both regular and super tankers. These times are not affected by the presence or absence of preemptions in the two versions of the model but reflect the values generated by the random number generator used by the simulator. The behavior of the two simulations causes the stream of random numbers to be applied to different events, resulting in two different sets of statistics.

## 5 DISCUSSION OF RESULTS

The results from the intervention model demonstrate that DELIM works correctly. The results are identical because the random number stream is generated in the same sequence every time it is used, and the random numbers affect the time of every event in the model. If only one event is not matched in the standard and the DELIM versions of each model, the entire result will change. Therefore, identical results imply that exactly the same chain of events took place in each version of the intervention model.

The situation for the anticipation model is different because it was presumed that the results of the two versions of this model would not be the same. To verify that DELIM works correctly in this model, it can be shown using a trace of actual model events that DELIM operates correctly in all possible circumstances. These circumstances are limited in number. The first occasion where DELIM is invoked in this model is when a regular tanker arrives in Seattle, the regular tanker dock is occupied and the super-tanker dock is available. DELIM should return 0 if no super-tanker is expected and a rule number otherwise.

| Type of Statistic | Standard Model | DELIM Model |
|---|---|---|
| Average round trip time: | | |
| Total | 14.00928 | 13.78273 |
| Regular tankers | 13.47393 | 13.22002 |
| Super tankers | 16.72920 | 16.64987 |
| Alaska dock utilization | 0.23866 | 0.22733 |
| Number of ships visiting port: | | |
| Regular tankers | 327 | 333 |
| Super tankers | 65 | 66 |
| Average loading time: | | |
| Regular tankers | 3.05850 | 2.85608 |
| Super tankers | 4.71806 | 4.45865 |
| Seattle regular dock utilization | 0.53327 | 0.51237 |
| Number of tankers visiting dock | 267 | 255 |
| Average unloading time | 0.72908 | 0.73381 |
| Average wait time | 0.49929 | 0.55783 |
| Maximum queue size | 3 | 4 |
| Seattle super-dock utilization | 0.53701 | 0.57879 |
| Number of tankers visiting dock: | | |
| Regular | 53 | 70 |
| Super | 64 | 65 |
| Average unloading time: | | |
| Regular | 0.72764 | 0.73101 |
| Super | 2.46042 | 2.46473 |
| Number of preemptions | 11 | N/A |
| Average wait time | 1.25828 | 1.23983 |
| Maximum queue size | 2 | 1 |

Figure 8: Anticipation model results

Figure 9 shows the event sequences from the simulation program which confirm the correct operation of DELIM for these two cases. When no super-tanker is expected, tanker T2 proceeds to the super-tanker dock because the regular dock is occupied. The other sequence of events shows that tanker T6 enters the queue on arrival when the regular dock is busy because DELIM indicates that a super-tanker is expected. The super-tanker arrival takes place 0.5 time units after the arrival of tanker T6.

```
No super-tanker expected          Super-tanker expected

Ship   Event    Time              Ship   Event    Time
ST1    departs  20.0              ST1    departs  20.0
(super-dock free)                 (super-dock free)

T1     arrives  22.4              T7     arrives  26.7
(regular dock busy)               (regular dock busy)
T2     arrives  22.5              T6     arrives  27.1
(DELIM called - returns 0         (DELIM called - returns 1
 T2 occupies super-dock)           T6 enters queue)

T1     departs  23.1              T7     departs  27.4
T2     departs  23.2              (T6 occupies regular dock)

                                  ST2    arrives  27.6
                                  T6     departs  28.1

                                  ST2    departs  30.0
```

Figure 9: Trace of arrival event sequence

DELIM is also invoked whenever a ship, either a regular or a super-tanker, leaves the super-dock. As shown by the event trace in figure 10, if no super-tanker arrival is expected when super-tanker ST2 departs, a regular tanker, T3, is dequeued and unloads at the super-tanker dock. Conversely, when tanker T10 leaves the super-dock, DELIM returns rule number 2, and the super-dock remains free. Super-tanker ST2 arrives 0.1 time units after T10 departs.

These event traces not only demonstrate that DELIM works correctly; they also show that the sequences of events in the DELIM version of this model occur as they would have occurred in the real world. So, for the anticipation model, the DELIM version is a more accurate representation of the real world and therefore results in a more valid simulation than the standard version.

```
No super-tanker expected          Super-tanker expected

Ship   Event    Time              Ship   Event    Time
ST1    arrives  30.7              T9     arrives  85.9
                                  (regular dock busy)
ST2    arrives  31.4
(enters queue)                    T10    arrives  86.1
                                  (DELIM called - returns 0
ST1    departs  33.3               T10 occupies super-dock)
(ST2 occupies super-dock)
T2     arrives  35.2              T9     departs  86.6

T3     arrives  35.3              T10    departs  86.8
(enters queue)                    (DELIM called - returns 2)

T4     arrives  35.4              ST2    arrives  86.9
(enters queue)

ST2    departs  35.7
(DELIM called - returns 0
 T3 occupies super-dock)
T2     departs  36.0
(T4 occupies regular dock)
T3     departs  36.3
(DELIM called - returns 0
 super-dock free)
T2     departs  36.8
(both docks empty)
```

Figure 10: Trace of departure event sequence

## 6 FUTURE RESEARCH

The next logical step in the investigation of DELIM is to compare the standard and DELIM implementations in a wider variety of models. The models should test variations of the timelines and different scenarios as well as combining two timelines in the same model. The application of DELIM in some simulation models of real world situations would be a practical test of DELIM's contribution to enhancing model validity. Actual measurements taken from real systems would provide an empirical basis for further conclusions regarding DELIM's efficacy.

Currently, DELIM is applied exclusively to discrete models of the event scheduling world view. A useful extension of DELIM research would be to investigate whether DELIM is applicable to simulations with other world views, such as activity scanning or process interaction. DELIM might also be employed in models which combine continuous and discrete modeling methods such as DMOD (Narain and Rothenberg 1989, 1990).

## 7 CONCLUSION

This research demonstrates that DELIM expands the capability of discrete event simulation without increasing the complexity of the models. The use of DELIM for the model in which the anticipation timeline occurs increases the validity of the model by permiting behavior which more closely resembles real world system behavior. By producing a more valid model, DELIM has expanded the capability of standard discrete event simulation.

DELIM is feasible. It is not difficult to incorporate DELIM into simulation programs, and the model objects within a simulation remain more general because the objects do not need specialized methods to handle the sequences of events depicted in the timelines.

DELIM's rules are easy to formulate. The correct formation or adjustment of a rule can be accomplished quickly, because a series of events which a rule represents are a natural way to express temporal relationships.

Finally, the complexity of simulation with DELIM is not increased vis-a-vis standard simulation models for the example models in this project. Although the models used to demonstrate DELIM are to some extent 'laboratory' models, DELIM offers practical and interesting possibilities for alternative discrete event simulation methodology. Further research into the use of DELIM in more complicated and realistic models which exhibit multiple timelines is certainly warranted.

## REFERENCES

Charniak, E. and D. McDermott. 1984. *Introduction to Artificial Intelligence*. Reading, MA: Addison-Wesley Publishing Company.

El Sheikh, A.A.R., R.J. Paul, A.S. Harding, and D.W. Balmer. 1987. A microcomputer-based simulation study of a port. *J. Opl. Res Soc.* 38:673-81.

Legge. G. 1992. *A Mechanism for Facilitating Temporal Reasoning in Discrete Event Simulation.* Ph.D. Dissertation, Department of Computer Sciences, The University of North Texas, Denton, Texas.

Legge, G. and D.L. Wyatt. 1991. A role for artificial intelligence in discrete event simulation. In *Proceedings of the 1991 Summer Computer Simulation Conference*, Baltimore, July 22-24, 400-404.

Narain, S. and J. Rothenberg. 1990. Proving temporal properties of hybrid systems. In *Proceedings of the 1989 Winter Simulation Conference*, New Orleans, December 9-12, 250-256.

Narain, S. and J. Rothenberg. 1989. A logic for simulating discontinuous systems. In *Proceedings of the 1989 Winter Simulation Conference*, Washington, D.C., December 4-6, 692-701.

Pritsker, A.A.B. 1974. *The GASP IV Simulation Language*. New York: John Wiley & Sons.

Stroustrup, B. 1986. *The C++ Programming Language.* Reading, MA: Addison-Wesley Publishing Company.

## AUTHOR BIOGRAPHIES

**GAYNOR LEGGE** is a recent graduate of the University of North Texas where she received M.S. and Ph.D. degrees in Computer Science. Her research interests include simulation, artificial intelligence and object oriented systems.

**DANA L. WYATT** is currently an Assistant Professor of Computer Science at the University of North Texas. She received a B.A.S.S. in 1978 and an M.S. in 1979 from Stephen F. Austin State University and a Ph.D. from Texas A&M University in 1986. Her research interests include simulation and database systems.