Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

SIMULATION GRAPH DUALITY: A WORLD VIEW TRANSFORMATION FOR SIMPLE QUEUEING MODELS

Lee Schruben
School of Operations Research
and Industrial Engineering
Cornell University
Ithaca, New York 14853, USA

Enver Yucesan
INSEAD
Boulevard de Constance
77305 Fontainebleau Cedex
FRANCE

ABSTRACT

Planar graphs play an important role in real world applications, partly due to the fact that some practical problems can be efficiently solved for planar graphs while they are intractable for general graphs. Simulation Graph Models of simple queueing systems are planar graphs. Their geometric duals can then be constructed. In the context of queueing models, this dualization process represents a transformation from the event scheduling to the activity scanning world view in discrete event simulation. The so-called primal-dual pair of models provides an alternative but equivalent representation of these stochastic systems.

1. INTRODUCTION

A graph is said to be *planar* if it can be drawn on the plane so that no two edges intersect except possibly at a vertex. A property of planar graphs is that they have *geometric duals*. In this paper, we show how this property can be used together with Simulation Graphs to define a transformation between two world views in discrete event simulation. A *Simulation Graph* is a network used to construct and analyze discrete event simulation models [Schruben and Yucesan, 1987]. On this network, each vertex represents state changes associated with a system event, while each directed edge depicts the logical and temporal relationships between these events. *World views* refer to system structuring approaches commonly used in discrete event simulation modeling [Schruben, 1983].

2. WORLD VIEWS

In discrete event simulations, world views provide alternate approaches to organizing a specification of model behavior. These modeling perspectives are commonly called *event scheduling*, *activity scanning* and *process interaction*. Even though each of these world views is supported by one or more simulation programming languages, no generally accepted definitions exist for any of the approaches. This is partly due to the fact that this classification scheme is neither mutually exclusive nor collectively exhaustive. A comprehensive discussion of world views can be found in [Nance, 1981], [Overstreet, 1987] and [Balci, 1988].

We next discuss the event scheduling and activity scanning world views in more detail. Our exposition is based on [Balci, 1988].

2.1 Event Scheduling Approach

In this world view, an event is the major focus for modeling a system. Within this approach, the objects in a system are identified first and described through state variables. Next, what causes changes in the values of state variables are defined as an event. Each event is usually implemented as a separate subroutine or procedure in the computer program.

The execution of such a model is carried out using an *events list*, a list of events scheduled to occur in the simulated future, and a global simulation clock. Each entity of the events list is called an *event record*, which contains information about the event type, event time and other attributes associated with the particular event.

The execution starts with initialization. The process continues by advancing the simulation clock to the time of the next (most imminent) event and the execution of the associated event routine. The latter may alter the values of state variables and may schedule further events by inserting

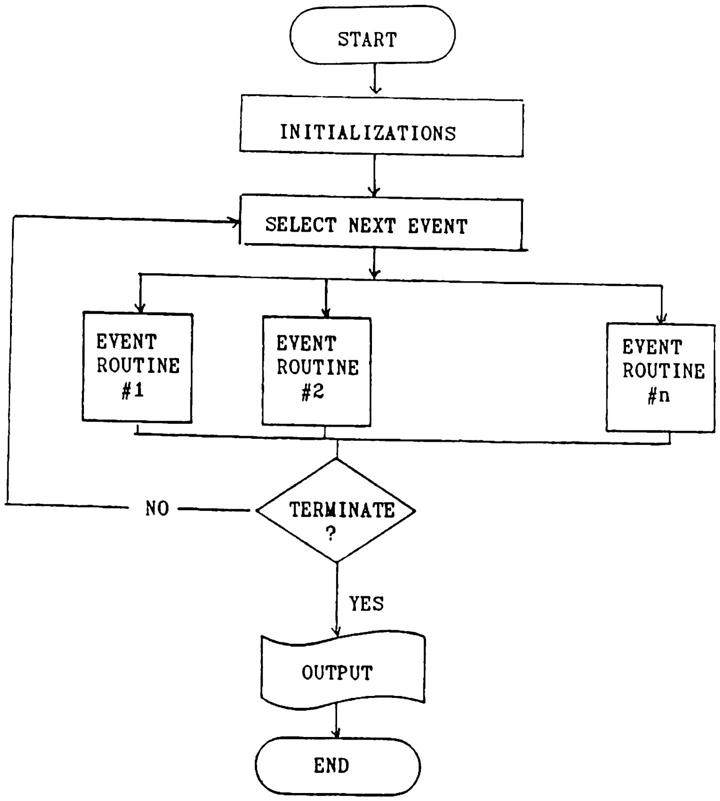


FIGURE 1. EVENT-SCHEDULING WORLD VIEW

appropriate event notices into the events list. The logic of the event scheduling approach is summarized in Figure 1.

2.2 Activity Scanning Approach

This approach is attractive for models where the number of possible events is small, but the conditions under which the events occur are very complex. Within this world view, the modeler describes an activity in two parts: (i) a condition which must be satisfied for the activity to take place, (ii) the operations of the activity performed upon the satisfaction of the activity's condition.

The logic of execution is as follows: the initializations include the assignment of initial values to state variables. The simulation clock is updated in fixed increments. That is, in Phase 1, time is advanced from t to (t+δt). Phase 2 is then conducted with a clock time of (t+δt). At that point, the conditions of activities are tested in the order of activity priorities. If an activity's condition is satisfied, the associated operations are performed. All the conditions must be repeatedly tested until no condition is satisfied at the current clock time. The activity scanning approach is summarized in Figure 2. Various modifications have been incorporated into this world view to enhance its execution efficiency. Due to space limitations, those modifications will not be discussed here.

3. SIMULATION GRAPHS

A Simulation Graph is a structure of the objects in a discrete event system that facilitates the development of correct simulation models. Events are represented on the graph as vertices. Each vertex is associated with a set of changes to state variables. The latter are used to describe various components of the system under study. Relationships between events, on the other hand, are represented as directed edges between pairs of vertices. Each edge depicts under what conditions and after how

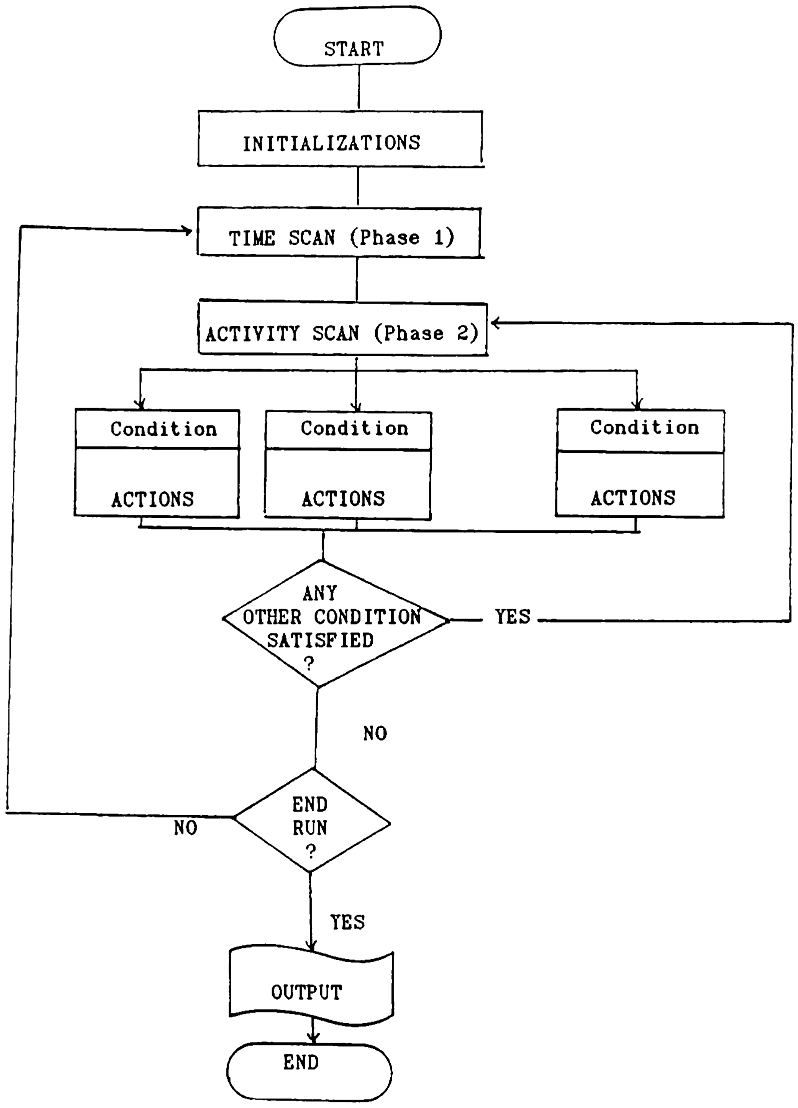


FIGURE 2. ACTIVITY SCANNING WORLD VIEW

much of a time delay an event will schedule or cancel another event.

A simple example is presented next to illustrate the involved concepts. For a complete treatment of Simulation Graphs, the reader is referred to [Schruben and Yucesan, 1987].

Example: Single Server Queueing System

We will develop a Simulation Graph Model of a single server queueing system. Suppose that customers arrive into the system every t_A time units and it takes the server t_S time units to attend to each customer. The state variables used in this model are:

Q, the number of customers waiting for service, and

S, the status of the server (0 = busy, 1 = idle).

The edge conditions for the model are:

(i) (The server is idle) $S = 1$,

(ii) (Customers waiting to be served) $Q > 0$.

The event descriptions are presented in Table 1.

The associated Simulation Graph is presented in Figure 3.

TABLE 1

Event Type	Event Description	State Changes
ARV	Customer arrival	Q = Q + 1

BGN	Beginning of service	S = 0
		Q = Q - 1

END	End of service	S = 1

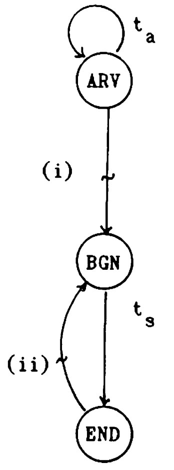


FIGURE 3. SINGLE-SERVER QUEUEING MODEL

4. GEOMETRIC DUALITY

Recall that a graph is said to be planar if it can be drawn on the plane so that its edges intersect only at their ends. A planar graph embedded in the plane is referred to as the plane graph. The "regions" defined by a plane graph G are called the faces of G, the unbounded region being the exterior face. Given a plane graph G, the geometric dual G^* is constructed as follows: Place a vertex in each face of G, including the exterior face. If two faces of G have an edge e in their common boundary, join the vertices of the corresponding faces by an edge e^* crossing only e. Note that the exterior face can have a boundary with itself. In digraphs, the direction of the edges in the dual graph can be assigned using the right hand rule.

The result may be a plane graph with loops or multiple edges. The graph G^D for which G^* is a plane graph is said to be the *dual* of G. The plane graph G is not unique; hence, its dual is not unique either. Figure 4 illustrates the dualization procedure.

When G is connected, the dual of the dual of G is isomorphic to G, that is $G^{**} \cong G$ [Nishizeki and Chiba, 1987].

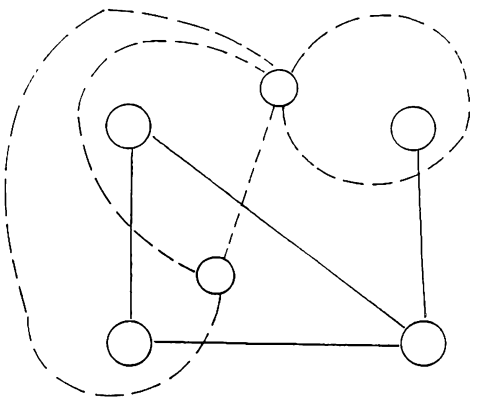
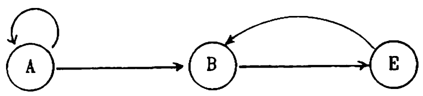


FIGURE 4. CONSTRUCTION OF G^* FROM G

5. DUALITY IN SIMULATION GRAPHS

5.1 Planarity

The following is a Simulation Graph for a generic queueing system such as a G/G/s model:



This generic Simulation Graph can represent many simple queueing systems. For example, different customer classes can be handled through the use of event attributes [Schruben and Yucesan, 1987]. That is, the arrival of a customer of type j can be represented by the event vertex A(j). The reader can easily verify that balking and blocking can also be incorporated without losing planarity. A proof of the foregoing assertions is presented in [Schruben and Yucesan, 1989].

Furthermore, it is possible to determine whether a given Simulation Graph is planar. There are a number of algorithms for establishing the planarity of graphs. For instance, one such algorithm is presented in [Hopcroft and Tarjan, 1974].

From this point on, we will only consider planar Simulation Graph Models.

5.2 Simulation Graph Duality

Since Simulation Graphs are planar, their dual can be constructed. As we will see shortly, the dual of a Simulation Graph is also a Simulation Graph. We will refer to $\mathcal{G}$ as the *primal Simulation Graph* or, simply, the *primal*; and $\mathcal{G}^D$ will be referred to as the *dual Simulation Graph* or, simply, the *dual*.

In this section, we will establish the fact that the primal Simulation Graph represents the event scheduling world view whereas the dual Simulation Graph represents the activity scanning world view. Hence, the process of constructing $\mathcal{G}^D$ from $\mathcal{G}$ really represents a transformation from event scheduling to activity scanning.

In this procedure, we will assume that the Simulation Graph does not contain any loops; that is, there are no edges that originate and terminate at the same event vertex. This is not a restrictive assumption since any loop can be replaced by a directed cycle. Figure 5 shows the single server queueing model under this convention. The dualization procedure is discussed next.

Step 0: Replace all the loops, if any, in the Simulation Graph with a directed cycle as described above.

Step 1: Ignore the directions on the edges and construct the geometric dual as before. The vertex in $\mathcal{G}^D$, corresponding to the exterior face of $\mathcal{G}$, is a special vertex. Label it "ESF" for "Event Scheduling Function."

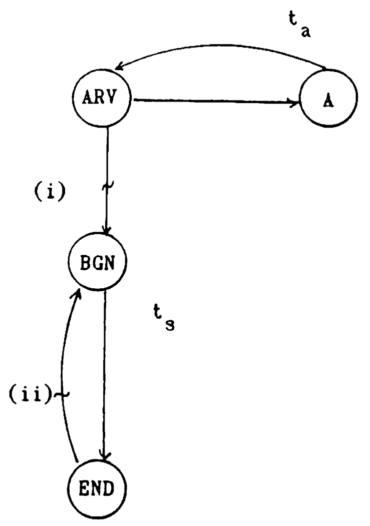


FIGURE 5. SINGLE-SERVER QUEUE REDRAWN

Step 2: Assign Edge Directions: There must be one edge directed from the ESF vertex to every other vertex in $\mathcal{G}^D$. All of the remaining edges should point in the opposite direction; that is, from a given vertex to ESF. Also note that the ESF vertex will always have a self-scheduling edge.

Step 3: Edge Conditions: In a Simulation Graph Model, there are two basic types of events. Events that are scheduled to occur unconditionally at specific instants in simulated time will be called *time-dependent* events or *t-events*. There are also those events which are scheduled only if the associated conditions are satisfied. These will be referred to as *conditional* events or *c-events*.

In $\mathcal{G}^D$, a new state variable is defined to condition the actions associated with a t-event in $\mathcal{G}$. The new state variable will basically denote the execution time of the associated activity. (An example will soon follow.)

No new state variables are defined in $\mathcal{G}^D$ for activities corresponding to c-events in $\mathcal{G}$. A condition, however, is assigned to the edge that is directed from ESF to the particular vertex. This condition is a complex one constructed by combining the edge conditions on the boundary of the corresponding face in $\mathcal{G}$ as well as those on any incoming edges through the Boolean operator AND.

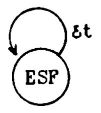
Step 4: Edge Delay Times: All edges directed from ESF to other vertices have zero delay times. The delay times in $\mathcal{G}^D$ will be assigned to those edges incident into the ESF vertex from a vertex that corresponds to a face defined by c-events in $\mathcal{G}$. The delay time will be equal to the sum of the delay times on the edges comprising the boundary of the corresponding face. For t-events, on the other hand, see Step 5.

Step 5: State Changes: The state changes associated with c-events in $\mathcal{G}$ directly carry over to the corresponding activities in $\mathcal{G}^D$. For the activities in $\mathcal{G}^D$ corresponding to t-events of $\mathcal{G}$, there is one additional state variable change: the update of the activity's execution time.

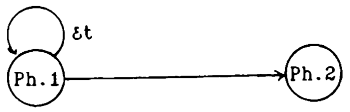
Step 6: Attribute Passing: The values of state variables are altered upon the execution of an activity. The new values of the state variables should be transmitted back to the ESF vertex using the attribute lists of the edges directed into ESF.

Step 7: Priorities: Priorities must be assigned by the modeler for activity execution (that is, for condition checking).

Step 8: Subdivision of ESF: The ESF vertex is subdivided into two vertices connected by a directed edge of zero delay time. The new vertices are called "Phase 1" and "Phase 2, respectively. The ESF vertex subdivided in this manner to conform to the activity scanning logic. More specifically, the ESF vertex



is replaced by:



Here, δt represents the fixed time increment that updates the simulation clock. Both of these vertices are always scheduled with the highest execution priority in $\mathcal{G}^D$.

This transformation is illustrated on the single server queueing model.

Example: (Revisited):

Figure 6 depicts the Simulation Graph for the single server queueing model along with its geometric dual. In the original (primal) model, since the arrival event (ARV) is a t-event, a new state variable, ARVT, is defined to denote the execution time of the arrival activity in $\mathcal{G}^D$. The edge conditions used in the dual are as follows:

(i) (Time for an arrival) ARVT = τ (τ denotes the simulation clock),

(ii) (Customers are waiting AND the server is idle) (Q>0) & (S=1).

The associated activities are described in Table 2.

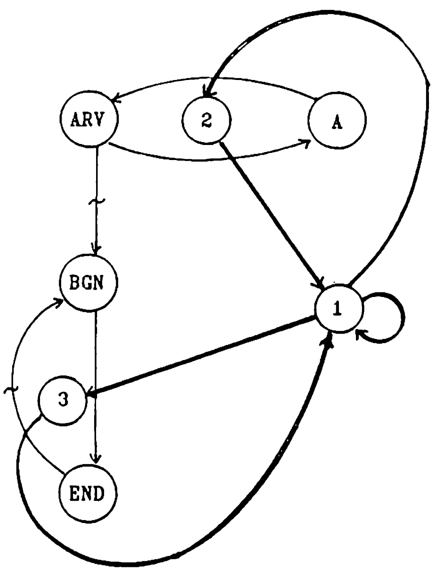


FIGURE 6. $\mathcal{G}$ AND $\mathcal{G}^D$

$\mathcal{G}$ is presented in Figure 7. Note that the execution of the dual model directly follows the activity scanning logic depicted in Figure 2.

TABLE 2

Activity Type	Activity Description	State Changes
PH.1	Phase 1: Time Scan	
PH.2	Phase 2: Condition Scan	Attribute List: Q, S, ARVT
ARV	Arrival Process	$Q = Q + 1$ $ARVT = \tau + t_A$
SVC	Service Process	$S = 0$ $Q = Q - 1$

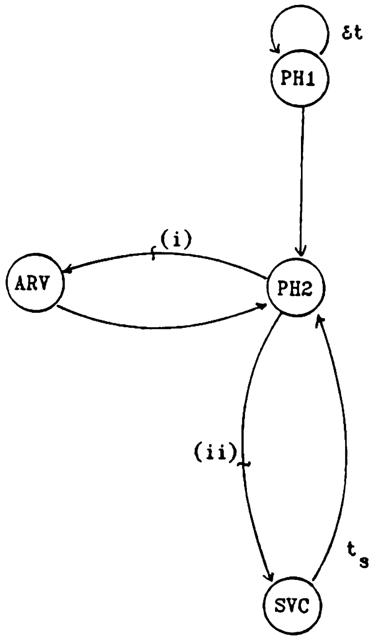


FIGURE 7. THE DUAL MODEL

6. CONCLUDING REMARKS

Within the context of simple queueing models, the dualization of Simulation Graphs represents a transformation from the event scheduling to the activity scanning world view. Hence, the primal-dual pair provides the modeler with alternative but equivalent representations of queueing systems.

We conjecture that this primal-dual relationship is valid not only for simple queueing systems, but for any discrete event dynamical systems in general.

Our procedure has several shortcomings. The first one is a direct consequence of the definition of the geometric dual. For instance, it is possible for different drawings of the same graph to result in different duals. Moreover, different graphs may have the same dual. More importantly, however, the dual of the dual model may not necessarily be isomorphic to the primal (original) model. This is because the transformation used in the dualization procedure is a one-to-many mapping.

As Hooper (1986) notes: "for more than a quarter of a century, simulationists in the U.S. and in Britain have been on 'divergent paths' as to world views (i.e., strategies). It does not appear that we are likely to alter these courses in the forseeable future. However, we can certainly benefit from a good understanding of the strategies that are in use." It is hoped that the construction presented in this paper provide a useful first step in bridging these paths. A potential benefit of this cooperation would be the new perspective it provides researchers, who have been studying simulation problems from a single point of view. Currently, research is under way for implementing some of the output analysis techniques, that have traditionally been studied in an event scheduling environment, using the activity scanning approach.

REFERENCES

[1] Balci, O. (1988) *The Implementation of Four Conceptual Frameworks For Simulation Modeling in High-Level Languages* Proceedings of the 1988 Winter Simulation Conference (Abrams, Haigh and Comfort, eds.) San Diego, CA.

[2] Hooper, J.W. (1986) *Activity Scanning and the Three-Phase Approach* Simulation Vol.47.5

[3] Hopcroft, J. and Tarjan, R. (1974) *Efficient Planarity Testing* Journal of the ACM Vol. 21.4 pp. 549-568

[4] Nance, R.E. (1981) *The Time and State Relationships in Simulation Modeling* Communications of the ACM Vol. 24.4

[5] Nishizeki, T. and Chiba, N. (1987) *Planar Graphs: Theory and Algorithms* North-Holland Mathematics Studies #140

[6] Overstreet, C.M. (1987) *Using Graphs to Translate Between World Views* Proceedings of the 1987 Winter Simulation Conference (Thesen, Grant and Kelton, eds.) Atlanta, GA.

[7] Schruben, L.W. (1983) *Simulation Modeling with Event Graphs* Communications of the ACM. Vol.26.11

[8] Schruben, L.W. and Yucesan, E. *On the Generality of Simulation Graphs* Technical Report #773, School of OR&IE, Cornell University. Ithaca, New York.

[9] Schruben, L.W. and Yucesan, E. (1989) *Simulation Graph Duality: A World View Transformation for Simple Queueing Models* Technical Report #842, School of OR&IE, Cornell University. Ithaca, New York.

AUTHORS' BIOGRAPHIES

LEE W. SCHRUBEN is a professor in the School of Operations Research and Industrial Engineering at Cornell University. He received his undegraduate degree from Cornell University, a Masters degree from the University of North Carolina, and a PhD from Yale University. Prior to coming to Cornell he was the Associate Director of the Health Systems Research Division of the Medical School at the University of Florida. His research interests are in the statistical design and analysis of large scale simulation experiments. His consulting activities have been primarily in the area of manufacturing systems simulation.

Lee W. Schruben
S.O.R.I.E
Upson Hall
Cornell University
Ithaca, New York 14853, USA
(607) 255 9128

ENVER YUCESAN is an assistant professor of Operations Management at the European Institute of Business Administration. He received his undergraduate degree in Industrial Engineering from Purdue University, a Masters degree and a PhD in Operations Research from Cornell University. His research interests include issues related to construction and structural analysis of discrete event models as well as statistical analysis of simulation output.

Enver Yucesan
INSEAD
Boulevard de Constance
77305 Fontainebleau Cedex, FRANCE
(33 1) 60 72 40 00