Proceedings of the 1993 Winter Simulation Conference
G.W. Evans, M. Mollaghasemi, E.C. Russell, W.E. Biles (eds.)

ACTIVITY CYCLE DIAGRAMS AND THE THREE PHASE METHOD

Ray J. Paul

Department of Computer Science
Brunel, The University of West London
Uxbridge, Middlesex UB8 3PH, United Kingdom

ABSTRACT

This tutorial paper looks at two modeling tools that are popular in European simulation modeling. These are Activity Cycle Diagrams, which represent the logical flow of a simulation model conceptually; and the Three Phase Method, which is a world view of program construction. Each tool is described, and comments made on their relative merits to the wider simulation community.

1 INTRODUCTION

Activity Cycle Diagrams and the Three Phase Method are two popular tools used in simulation modeling in Europe, and in particular in the United Kingdom. Whilst some simulationists use both tools in their work, it is worth observing that the two tools can be used independently, as will be shown. Activity Cycle Diagrams (ACD, sometimes called Entity Cycle Diagrams, or various combinations of Entity, Activity and Cycle Diagrams) are a method of conceptualizing the problem in terms of the logical flow of objects in the system.

An ACD can be used as a basis for a model written in object-oriented code, or any program using any of the event, process, or three-phase world views. However, some simulation packages, especially simulators, require a representation of the problem to be modelled that does not map directly to an ACD. Even in such cases, an ACD can be used for problem understanding, prior to modeling. ACDs are discussed in the second section of this paper.

The Three Phase Method, or World View, is a competitor to the more well known event and process world views (Law and Kelton, 1991). The method grew out of the activity based approach popular in the United Kingdom in the 1960s. A description of the Three Phase Method is given in the third section of the paper, and some comments on its wider applicability in the fourth section.

The technical contents of this paper are taken from Paul and Balmer (1993). Another text that discusses this is Pidd (1992a) with some programming and other supporting material covered in Pidd (1989).

2 ACTIVITY CYCLE DIAGRAMS

2.1 Basic Concepts

Activity Cycle Diagrams (ACDs) are one way of modeling the interactions of system objects and are particularly useful for systems with a strong queueing structure. They are based on Tocher's (1963) idea of stochastic gearwheels. ACDs have the advantage of parsimony in that they use only two symbols which describe the life cycle of the system's objects or entities:

An *entity* is any component of the model which can be imagined to retain its identity through time. Entities are either idle, in notional or real *queues*, or active, engaged with other entities in time consuming *activities*. The symbols we use are shown in Figure 1.

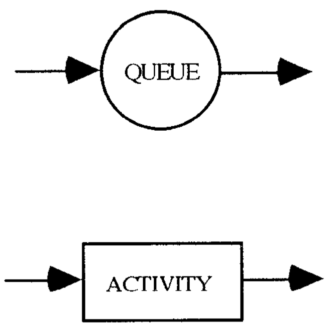

Figure 1: Queue State and Activity State

An active state usually involves the co-operation of different classes of entity. The duration of the active state can always be determined in advance - usually by taking a sample from an appropriate probability distribution if the simulation model is stochastic. For example, the unloading of a ship at a berth is an active state, where an entity ship and an entity berth are engaged in the activity unload (possibly with other entities as well, such as cranes etc.).

A passive state or queueing state involves no co-operation between different classes of entity and is generally a state in which the entity waits for something to happen. The length of time an entity will spend in a queue cannot be determined in advance because it depends on the duration of the immediately preceding and succeeding activities. For example, the time a ship spends waiting in an idle queue for unloading at a berth depends on its time of arrival and the time one of the berths it can use becomes vacant.

A life cycle (activity cycle) of queues and activities is defined for each entity type. We impose the restriction that queues and activities must alternate in any life cycle (if necessary we make this happen by creating dummy queues). A complete ACD consists of a combination of all the individual life cycles.

2.2 The Pub Example

We shall show how to draw an ACD using the Pub example. This example is used by many authors (e.g. Clementson, 1982) since its background is implicitly understood by most readers. The first simple version has three entities called 'man', 'barmaid' and 'glass'. The man either drinks or waits to drink. The barmaid either pours a drink or is idle. The glass is either used to drink from, is empty, is poured into by the barmaid or is full waiting to be drunk from. We can summarise the states for each entity as follows in Figure 2.

ENTITY	STATES	SYMBOL
Man	Drink	□
	Wait	○
Barmaid	Pour	□
	Idle	○
Glass	Drink from	□
	Empty	○
	Pour into	□
	Full	○

Figure 2: Pub States

Each life cycle for each entity type can then be drawn as in Figure 3.

The ACD for the pub is then drawn by combining the common activities as in Figure 4.

The ACD illustrates logically that the activity DRINK cannot start unless a man is in the queue WAIT and a glass is in the queue FULL. Similarly the activity POUR cannot start unless a barmaid is in the queue IDLE and a glass is in the queue EMPTY.

The ACD also has a stronger interpretation. This is, that when is a man in the queue WAIT and a glass in the queue FULL, then the activity DRINK will start. Similarly, when there is a barmaid in the queue IDLE, and a glass in the queue EMPTY, then the activity POUR will start.

On completion of any activity, the movement of the entities is fixed. After POUR, the barmaid goes to the queue IDLE, and the glass goes to the queue FULL. After DRINK, the glass goes to the queue EMPTY, and the man to the queue WAIT.

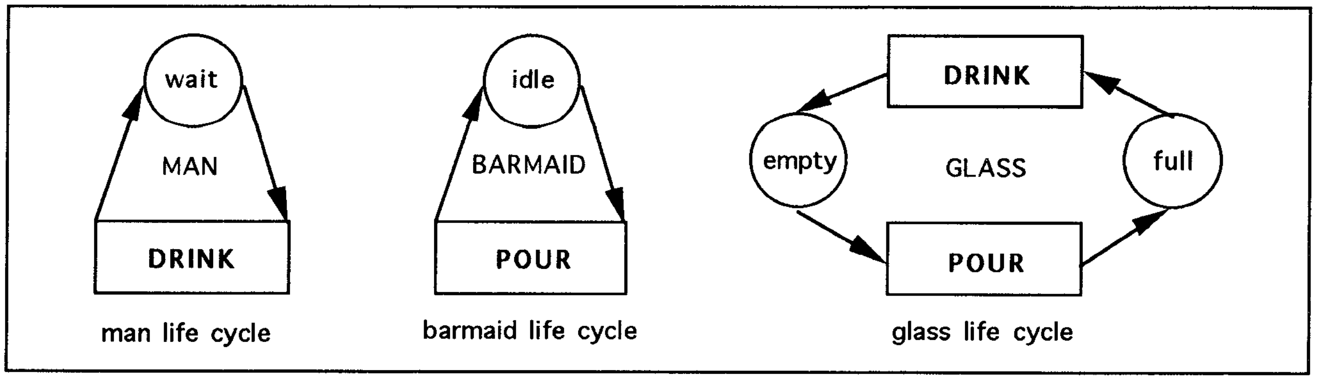

Figure 3: Pub Entity Cycles

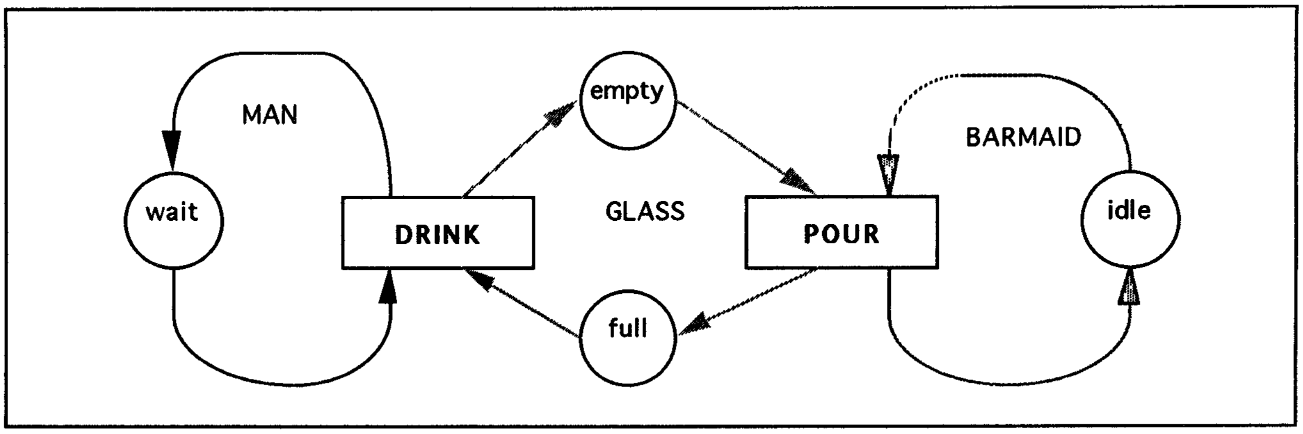

Figure 4: Pub ACD

It is a useful, but not essential, convention that queues and activities should alternate. This makes for more robust modeling in the event of inevitable change to the model requirement (as will be illustrated later). It is also a useful, but not essential convention, that all entity life cycles should be closed. Whilst this necessitates the introduction of an 'outside world' queue for entities that 'visit' the system, it has the advantage of assisting the analyst in thinking through the life cycles of the entities more rigorously, and hence with an increased prospect of success.

3 THE THREE PHASE METHOD

3.1 Manual simulation using the ACD

The first step is to make sure that the logic of the system is properly understood and one of the best ways of doing this is to run a manual simulation. There are a variety of methods for doing this, but we shall use the ACD method. This will help understand the Three Phase Method.

In order to carry out a manual simulation with an ACD, we draw the life cycles on a large sheet of paper or playing board (using different colours to distinguish the cycles of different entity types). The current state of the model is described by the position of each entity in a queue or activity; this is easily shown using coloured counters at appropriate points on the playing board.

An event is a change in the state of the model which occurs at an instant of time. When an activity starts, its duration can be sampled from a specified distribution, and the time when it will finish can be noted on the playing board or on the next event list. The activity is bound to finish at exactly that time, so the completions of activities are bound events. However we do not know in advance when an activity can start: this depends on the correct combination of entities being available in the preceding queues. The starts of activities are conditional events.

One of the benefits of manual simulation is to establish priorities where they exist. In the final pub example in Figure 7, the entity barmaid could face a possible choices of activity to start first. It may be important to establish that there is a priority and what it is. Writing computer code directly can easily result in this problem being forgotten and handled haphazardly.

3.2 The Pub Example Modified

In the Three Phase Method, the simulation proceeds as a repetition of the following three phases :

Phase 1
Check the finish times of all the activities currently in progress. Find the earliest of these. Advance the clock to this time.

Phase 2
For the activity (or activities) which have finished, move the entities into their appropriate queues. Cross out the note showing when the activity was to end.

Phase 3
Scan the activities in order of increasing activity number (they should have been numbered before the start of the run). Start any activities which can begin by moving the appropriate entities from the queues into the activity. Sample an activity duration time, calculate when the activity will finish, and make a note of this time.

Note this common simulation structure:

- Advance time to next event;
- Execute Bound events (activity completions);
- Execute Conditional events (activity starts).

We can record the state of the simulation using these three phases. So if we say that the activity drink takes 4

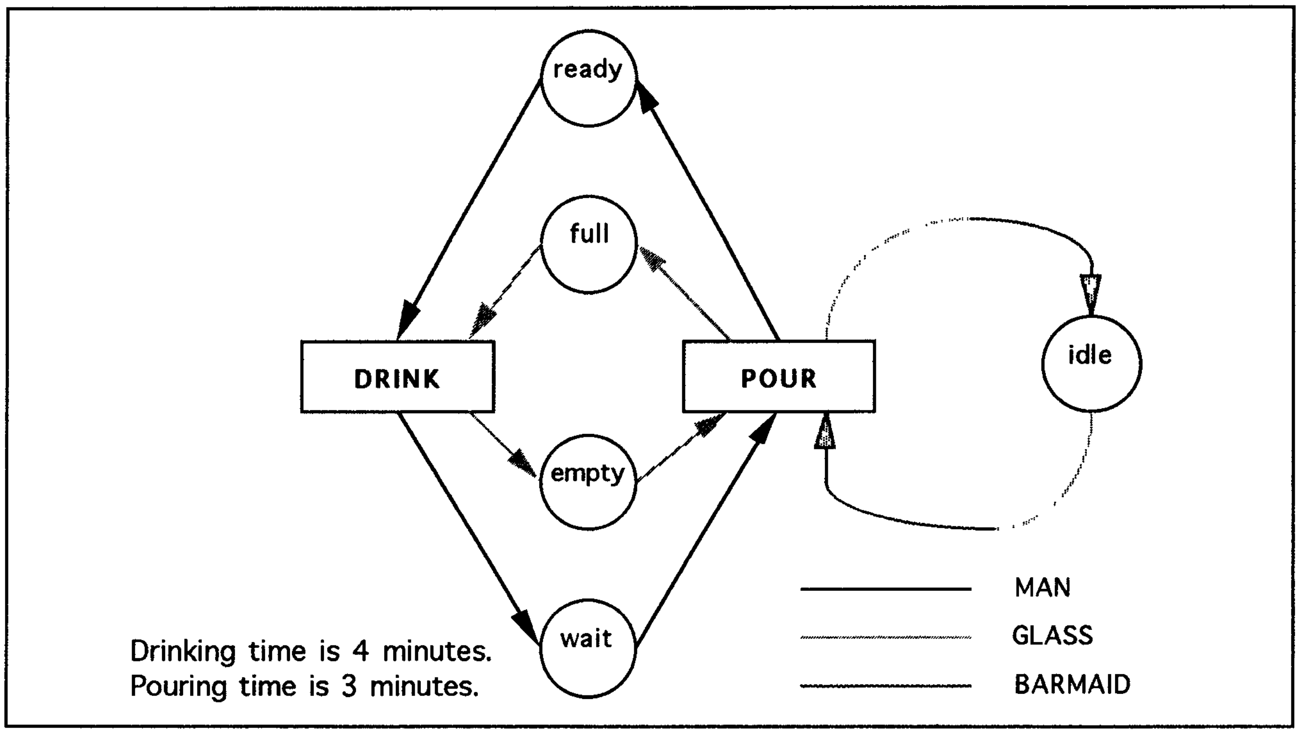

Figure 5: The Modified Pub Activity Cycle Diagram

minutes; pour takes 3 minutes; glasses and customers are synchronised as in Figure 5; and every entity in their appropriate starting queues IDLE, EMPTY and WAIT, then we get Table 1.

A	B	C
0	-	pour starts, ends at 3
3	pour ends	drink starts ends at 7 pour starts, ands at 6
6	pour ends	drink starts, ends at 10 pour starts, ends at 9
7	drink ends	-
9	pour ends	drink starts, ends at 13 pour starts, ends at 12
10	drink ends	-
12	pour ends	drink starts, ends at 16 pour starts, ends at 15

Table 1: Manual Simulation of Pub

We could of course collect statistics from a manual simulation, but this would be very tiresome so we use the computer to automate the whole process. Nevertheless, it is worth re-emphasising that a manual simulation is an important step in the understanding of the process being modelled.

3.3 The Three Phase Method

The three phases to be performed are usually expressed as A, B and C as we have seen. The executive cycles through the phases as the simulation proceeds.

A PHASE (time scan): determine when the next event is due and decide which B events are then due to occur. Move simulation clock time to the time of the next event.

B PHASE (B calls): execute only those B events identified in the A phase as being due now.

C PHASE (C scan): attempt each of the C events in turn and execute those conditions that are satisfied. Repeat the C scan until no more C events can take place (i.e. no more activities can start).

An outline flow diagram for such an executive is shown in Figure 6.

The three phase method or approach was first described by Tocher (1963). Its basic building block is an activity, which has two events that describe it, an end of activity event and a start of activity event:

B events are *bound* or book-keeping events signifying the end of an activity for an entity. They are executed directly by the executive program whenever their scheduled time is reached.

C events are *conditional* or co-operative events signifying the start of an activity for the relevant entities. They are executed because of the co-operation of different classes of entity or the satisfaction of specific conditions within the simulation.

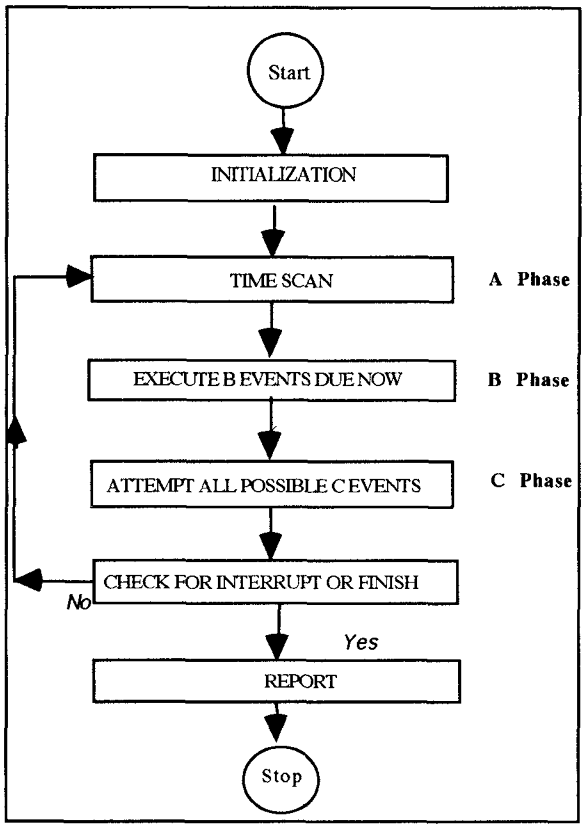

Figure 6: A Three Phase Executive

Each B and C event is programmed as an independent program routine or procedure. In effect then, they represent an 'atomistic' description of the activities of a simulation problem, where the start of an activity, a C event, is identified, tested for and set up if the test is successful.

The completion of an activity for each entity involved in it is separately identified in a B event and is executed at the scheduled time. It is the atomistic structure combined with an efficient executive that makes the three phase method so powerful. In summary, the method has the following desirable characteristics:

- Modeling clarity - reduces risk of error and aids verification.
- Model maintainability - an ongoing process with changing personnel.
- Modularity - a special technique to cope with very large models. Combining smaller ones can be incorporated more readily.
- Interaction - inclusion of optional gaming elements is easy.

The adoption of a good standard such as this method can provide benefits in a number of areas. These are:

- Total lifetime software costs can be controlled.
- Control of subcontract staff is easier.
- Staff mobility is enhanced.

4 WORLD VIEW COMPARISONS

4.1 Other world Views

Activity Based

The simplest structure is to go through all the activities, testing each in turn and starting or ending the appropriate ones. If any activity is executed, the whole list may need to be searched again. It is possible that an activity higher up the list, which was previously blocked, can now be executed under the new state of the system.

The chief advantage of this type of structure is that it is easy to program. Each activity can be programmed and tested as a separate module. The simplicity of the structure is particularly valuable when dealing with logically complicated models, where activities are predominantly "multi-resource", i.e. activities that require several different entities to be in particular states before they can start. An example of this is an activity for causing a ship to leave port, where the conditions might be that a ship is waiting to depart, a tug is available, a pilot is available, the tide is in and the dock entrance is free.

However, an activity-based structure produces an inefficient computer program, in the sense that a great many unsuccessful tests have to be made. Improvements can be made by only recycling if certain activities are executed, and not otherwise.

Purely activity-based languages are now more or less defunct, having been replaced by the Three Phase Method. The three phase structure combines a certain amount of efficiency in performing only the relevant bound event, with an ability to handle logically interrelated activities in a simple manner; all being tested on every iteration.

Event Based

A different approach is to make use of the information available from the time scan to branch directly to the relevant bit of program. This is an event-based structure. The executive branches directly to the activity associated with the earliest event identified by the Time Scan. The efficiency of this type of structure is its main virtue, but it can be very difficult to use in multi-resource type

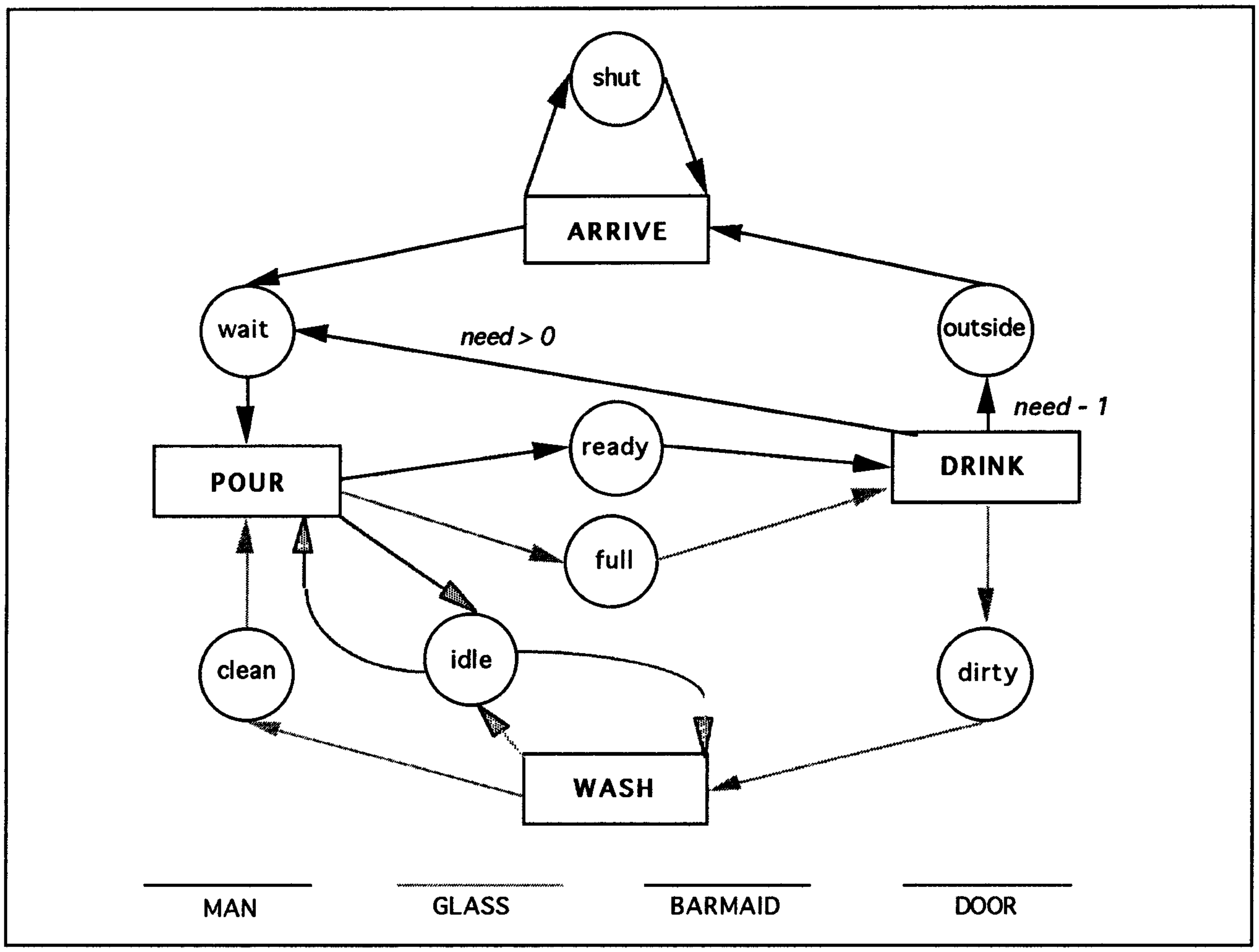

Figure 7: Complex Pub ACD

models, where cross-referencing between branches arise. A large number of possible states of the system have to be considered when writing the program.

The Process Flow Method
A simulation structure which is increasing in popularity is the process flow method based around the SIMULA simulation language. SIMULA is ALGOL 68 plus. The essence of the method is to write each entity life cycle in an ACD as a block of code with PAUSE and ENABLE commands to signify that the cycle cannot continue until other entities are available to enable an activity to take place. The executive of such a program requires complex and extensive cross referencing of the blocks of code and alteration to the model is difficult.

4.2 Simultaneous Events

One of the most difficult problems in setting up a simulation program is to cater for simultaneous events. The model will probably be used to make comparative runs, where it is important to ensure that unwanted differences between runs are not generated because of changes in the order in which activities are executed. This is also necessary for debugging the program by following the progress of the simulation in detail.

If we use the more complex version of the pub in Figure 7, then the three phase world view expresses priority as follows.

Three Phase

B events	C events
End Arrive	C1 - Start Arrive
End Pour	C2 - Start Pour
End Drink	C3 - Start Drink
End Wash	C4 - Start Wash

Priority ? : Pour before Wash.

Hence C2 is listed in the program before C4. Because all B events are completed after a time advance before moving to the C phase, all entities that will be made available at the time can be allocated by priority in the C phase.

Event

Events	Event notice posting (i.e. events generated)
E1 - Arrive	E1, E2
E2 - Pour	E2, E3, E4
E3 - Drink	E2, E4
E4 - Wash	E2, E4

Priority ? :

We have no control over the order in which the event notices are posted. We can of course use an attribute of an event notice to rank events. Is it obvious what the priorities are?

Process

Each process handles all activities and queues in the life cycle description of an entity in the ACD (or some equivalent description of the life cycle or process). Priorities are easily handled within processes, but the same priority problems can recur in different processes, or in different mixtures.

4.3 Amending Models

Changing models is an everyday occurrence (Paul, 1991). Let us assume that the beer in the pub comes from a barrel of a fixed size. When the barrel is empty, the barmaid needs to change it for another barrel. Let us call this activity FILL. One way of handling this would be to alter the ACD as in Figure 8.

The subsequent amendments to the model are as follows:

Three Phase

Add:

B event	C event
End Fill	C5 - Start Fill

Priority: Fill before Wash.
Fill generates its own priority.

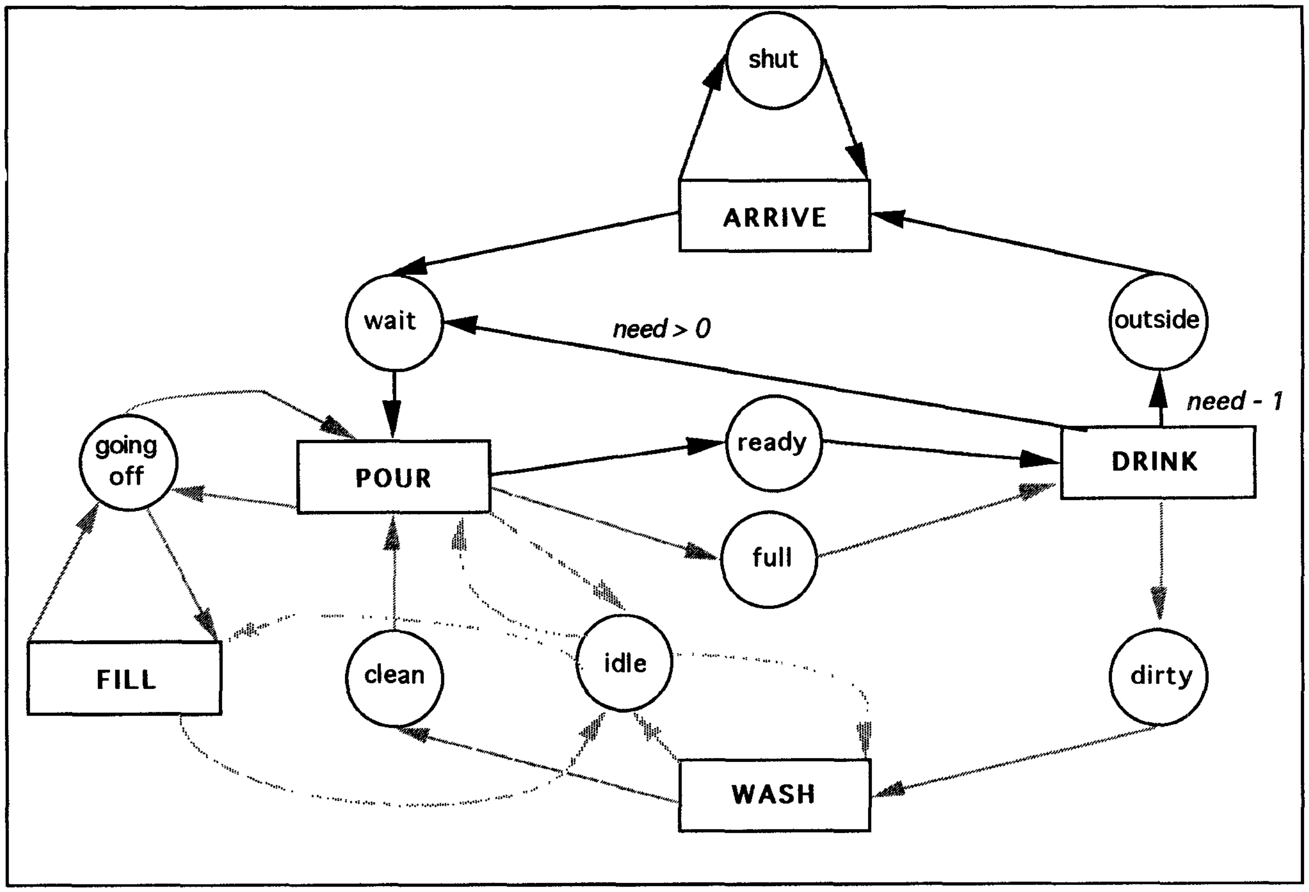

Figure 8: Complex Pub ACD Modified

Amendments:
For C2, Start Pour, add the condition that keg is greater than 0. If End Pour is generated, subtract 1 from keg.
Here amendments are local and obvious.

Event
Add:

Event	Event Notice Posting
E5 - End Fill	E2, E4

Amendments:
Each event needs checking. In fact E2 and E4 need to check for E5. Of course, one would set up the conditions and code so that E2 and E4 are subroutines to make programming efficient. But the essence is that it is not obvious without checking everything where changes should be made. But priority ?

Process
The introduction of the new keg process impacts on nearly all the other processes in the model (except the door). Each process has to be carefully checked, and all local priorities need to be carefully re-examined.

In *Conclusion*, all objections to the Event and Process methods can be overcome by separate resource routines, ingenuity, cunning, etc. All of these make for a non-standard structure and rely on the analyst to a greater extent. Three Phase is a safe standard

- it is easier to write
- it is easier to understand
- it is easier to validate (small blocks of independent program code)
- it is easier to change.

5 CONCLUSIONS

ACDs and the Three Phase Method have been successfully used in combination. For example, see El Sheikh et al (1987), Holder and Gittins (1989), Williams et al (1989) and Stapley and Holder (1992). These papers describe the use of ACDs with a particular Three Phase based simulation package which is described in Crookes et al (1986) and by Paul and Balmer (1993).

The Three Phase Method has much in common with production rules in expert systems, and this analogy is drawn out in Paul (1989) or Paul and Doukidis (1992). A particular method of handling excessive randomness in statistical sampling has been shown to be easily implemented in the Three Phase Method (Saliby and Paul, 1993). The Three Phase Method is particularly suited to automatic program generation because of its atomistic structure, and this is discussed by Mathewson (1982 and 1985), Paul and Chew (1987) and Paul and Balmer (1993).

ACDs are a particularly powerful conceptual modeling tool around which methods for automating model formulation have evolved (Paul and Doukidis, 1986). These and other applications are the basis of many developments by the Computer Aided Simulation Modeling group (Paul, 1992), whose origins are described by Balmer and Paul (1986).

Some support to the claims of Three Phase adaptability is provided by Holder and Gittins (1989) and Williams et al (1989). Two teams set about building different parts of the same problem using ACDs and the Three Phase Method. On completion of the two projects, developed by independent software houses for the same customer, the two models were successfully integrated together with little difficulty.

However, it would be churlish to end on a note of absolute triumph. ACDs are limited. They show logical flow, but not logical depth. Whilst it is inevitable that any representation method is either easy to follow and not comprehensive, or vice versa, sometimes the limitations can be severe: see El Sheikh et al (1987) for a simple example. And the Three Phase Method has yet to prove its adaptability in a fast changing world. Pidd (1992b) shows that Three Phase modeling and Object-Oriented thinking can be combined. Parallel simulation has yet to be shown to be compatible with the Three Phase Method, although it is too early to say that it cannot be done.

REFERENCES

Balmer, D. W. and R. J. Paul. 1986. CASM- The right environment for simulation. *Journal of the Operational Research Society*, 37:443-452.

Clementson, A. T. 1982. *Extended control and simulation language*, Birmingham: Cle. Com Ltd.

Crookes, J. G., D. W. Balmer; S. T. Chew; and R J. Paul. 1986. A Three-phase simulation system written in Pascal. *Journal of the Operational Research Society*, 37:603-618.

El Sheikh, A. A. R., R. J. Paul, A. S. Harding and D. W. Balmer. 1987. A Microcomputer based simulation study of a port, *Journal of the Operational Research Society*, 38:673-681.

Holder, R. D. and R. P. Gittins. 1989. The effects of warship and replenishment ship attrition on war arsenal requirements. *Journal of the Operational Research Society*, 40:155-166.

Law, A. M. and W. D. Kelton. 1991. *Simulation Modeling and Analysis*, New York: McGraw-Hill.

Mathewson, S. C. 1982. A DRAFT II/SIMON Manual, Management Science, Imperial College, London.

Mathewson, S. C. 1985. Simulation program generators: Code and animation on a PC. *Journal of the Operational Research Society*, 36:583-589.

Paul, R. J. 1989. Artificial intelligence and simulation modelling. In *Computer Modelling for Discrete Simulation* , ed. M. Pidd, London: Wiley.

Paul, R. J. 1991. Recent Developments in simulation modelling. *Journal of the Operational Research Society*, 42:217-226.

Paul, R. J. 1992. The computer aided simulation modeling environment: An overview. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldman, R. C. Crain and J. R. Wilson, Society for Computer Simulation, San Diego.

Paul, R. J. and D. W. Balmer. 1993. *Simulation modelling*. Stockholm: Chartwell-Bratt Student-Text Series.

Paul, R .J. and S. T. Chew. 1987. Simulation modelling using an interactive simulation program generator. *Journal of the Operational Research Society*, 38:735-752.

Paul, R. J. and G. I. Doukidis. 1986. Further developments in the use of artificial intelligence techniques which formulate simulation problems. *Journal of the Operational Research Society*, 37:787-810.

Paul, R. J. and G. I. Doukidis. 1992. Artificial intelligence and expert systems in simulation modelling. In *Artificial intelligence in operational research*, ed. G. I. Doukidis and R. J. Paul, 229-238, Basingstoke: Macmillan.

Pidd, M. (ed.) 1989. *Computer modelling for discrete simulation*. Chichester: John Wiley and Sons.

Pidd, M. 1992a. Computer simulation in management sciences. 3rd ed. Chichester: John Wiley and Sons.

Pidd, M. 1992b. Object orientation and three phase simulation. In *Proceedings of the 1992 Winter Simulation Conference*, ed. J. J. Swain, D. Goldman, R. C. Crain and J. R. Wilson, Society for Computer Simulation, San Diego.

Saliby, E. and R. J. Paul 1993. Implementing descriptive sampling in three phase discrete event simulation models. *Journal of the Operational Research Society*, 44:147-160.

Stapley, N. R., and R. D. Holder. 1992. The development of an amphibious landing model, *Journal of Naval Science* 18:193-202.

Tocher, K. D. 1963. *The art of simulation.*. London: English Universities Press.

Williams, T. M., R. P. Gittins and D. M. Burke. 1989. Replenishment at sea. *Journal of the Operational Research Society*, 40:881-887.

AUTHOR BIOGRAPHY

RAY J PAUL holds the first U.K. Chair in Simulation Modelling at Brunel University. He previously taught information systems and operational research for 21 years at the London School of Economics. He received a B.Sc. in Mathematics, and a M.Sc. and a Ph.D. in Operational Research from Hull University. He has published widely in book and paper form (two books, over 70 papers in journals, books and conference proceedings), mainly in the areas of the simulation modeling process and in software environments for simulation modeling. He has acted as a consultant for a variety of U.K. government departments, software companies, and commercial companies in the tobacco and oil industries. His research interests are in methods of automating the process of discrete event simulation modeling, and the general applicability of such methods and their extensions to the wider arena of information systems. Recent research results have been in automatic code generation, colour graphics modeling interfaces, dynamically driven icon representations of simulation models, machine learning applied to model specification and to output analysis, object oriented approaches, and information systems paradigms. He is currently running a simulation research group of three faculty members and eight research students. He has recently instituted the first M.Sc. in Simulation Modelling, a one year course starting in October each year at Brunel University.