GENERAL PURPOSE SYSTEMS FOR EFFECTIVE CONSTRUCTION SIMULATION

By Julio C. Martinez[1], M. ASCE, and Photios G. Ioannou[2], M. ASCE

ABSTRACT: This paper examines the characteristics of discrete-event simulation systems in terms of their application breadth (general or special purpose), modeling paradigm (process interaction vs. activity scanning), and flexibility (programmable or not). Several construction simulation systems are examined with primary emphasis on CYCLONE and STROBOSCOPE as representatives of the wide range of tools that are currently available. CYCLONE is a well-established, widely used and simple system that is easy to learn and effective for modeling many simple construction operations. STROBOSCOPE is a programmable and extensible simulation system designed for modeling complex construction operations in detail and for the development of special-purpose simulation tools. The characteristics of these systems, as well as other recent developments, illustrate that an effective general-purpose simulation tool for construction is in essence one based on extended forms of Activity Cycle Diagrams and the Activity Scanning modeling paradigm. As explained through several examples, these representations are indeed the most convenient and intuitive for construction simulation systems. Furthermore, the programmability of such a system is the principal factor that determines its power, flexibility, and ease of learning and use.

INTRODUCTION

Discrete-event simulation has been used to analyze and design construction operations for over three decades. This extensive research activity has created an implicit but limited notion of what constitutes a "construction simulation system". It has also created an implicit understanding that, conceptually, these systems are indeed better suited to construction operations modeling than systems commonly used in other industries. A clear and explicit explanation of the essence of a "construction simulation system" and the reasons for which these systems are a natural choice for construction modeling, however, are not available. This paper explains in detail the essence of a construction simulation system as well as the reasons that make such systems intuitively ideal for construction operations modeling.

This sets the background for a review of the various systems that have been designed for modeling construction operations. Although a representative set of tools is covered, the emphasis of this paper is on CYCLONE (Halpin & Riggs 1992) and STROBOSCOPE (Martinez 1996). CYCLONE is a well-established, widely used, and simple system that is easy to learn and effective for modeling relatively simple construction operations. STROBOSCOPE is a programmable and extensible simulation system designed for modeling complex construction systems in detail and for the development of special-purpose simulation tools.

SIMULATION TOOL CHARACTERISTICS

The scope, flexibility, and suitability-to-task of a simulation tool depends on its application breadth, modeling paradigm (simulation strategy) and flexibility.

Application breadth is the scope of models for which the tool is designed. *General-purpose* simulation tools and languages target a very broad domain and can be used to model almost any type of operation. General-purpose simulation tools for construction modeling include those described in (Halpin 1976, Chang 1986, Paulson et. al. 1987, Ioannou 1989, Mohieldin 1989, Liu 1991, Odeh 1992, Sagert 1995). In contrast, *special-purpose* simulators are tools that target a narrow domain such as ductile iron pipe installation. Special-purpose simulators designed for specific construction tasks include those described in (McCahill and Bernold 1993; Shi and AbouRizk 1997; Oloufa and Ikeda 1997; Martinez 1997, 1998).

Modeling paradigm or simulation strategy is the conceptual framework that guides model development and determines how the modeler views the system being modeled (Hooper 1986, Balci 1988). The two main simulation strategies are Process Interaction (PI) and Activity Scanning (AS). Event Scheduling (ES) is a third simulation strategy that is often combined with PI or AS. (Some authors also consider a *transaction-based approach* that is very similar but subtly different to PI. In the following, all statements about PI also apply to the transaction-based approach.)

Flexibility reflects the capability of the tool to model complex situations and to adapt to a wide range of application requirements. Advanced simulation systems typically involve computer programming and are flexible enough to model very complex operations in detail. Simpler non-programmable tools are typically easier to learn and can be used to model many simple operations effectively. However, they often require assumptions that prevent the effective analysis of many complex or detailed operations.

Simulation Strategy

By far the most significant characteristic of any general-purpose discrete-event simulation system is its *simulation strategy*. The main simulation system strategies in use today for modeling construction processes are Process Interaction (PI) and Activity Scanning (AS).

A Process Interaction model is written from the point of view of the entities (transactions) that flow through the system.

[1] Assistant Professor of Civil Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0105.
[2] Associate Professor of Civil & Environmental Engineering, University of Michigan, Ann Arbor, MI 48109-2125
Note. This manuscript was published in theASCE *Journal of Construction Engineering and Management*, Vol. 125, No.4, July/August 1999, pp. 265-276.

Table 1 – Activities, required conditions and outcomes for a scraper and pusher operation

Conditions Needed to Start	Activity	Outcome of Activity
Pusher at push-point Scraper at cut	PushLoad	Pusher ready to backtrack Loaded scraper ready to haul
Pusher ready to backtrack	Backtrack	Pusher at push-point
Loaded scraper ready to haul	Haul	Loaded scraper ready to dump
Loaded scraper ready to dump	DumpAndSpread	Dumped soil Scraper ready to return to cut
Scraper ready to return to cut	Return	Scraper at cut

These entities typically arrive, undergo some processing where they capture and release scarce resources, and then exit. This strategy is particularly suited to modeling operations where the moving entities are differentiated by many attributes and where the machines or resources that serve these entities have few attributes, a limited number of states, and do not interact too much (Hooper 1986). Most operations in manufacturing and the industrial and service industries are of this type. Consequently, a large number of commercial simulation tools are based on the Process Interaction paradigm (e.g., GPSS, SIMAN, SLAM, ProModel, SimScript, ModSim, Extend, etc.)

In contrast, Activity Scanning models are written from the point of view of the various activities that are performed and focus on identifying these activities and the conditions under which they take place. Three-Phase Activity Scanning is a modified approach that incorporates Event Scheduling concepts to increase performance (Tocher 1963). Activity Scanning is particularly adept at modeling systems with interdependent components subject to complex activity startup conditions where many resources with distinct properties must collaborate according to highly dynamic rules (Hooper 1986). Most Activity Scanning languages were developed in the sixties in Europe. They include GSP (Tocher and Owen 1960), CSL (Buxton and Laski 1962), and HOCUS (Hills 1971).

Simulation strategies are independent of other tool characteristics such as object-orientation or hierarchical decomposition. Simulation strategies also give special-purpose simulators their unique flavor and shape simulation model development even when using general-purpose programming languages (Balci 1988). The strategy used by a simulation tool, whether it is Process Interaction or Activity Scanning, has a strong impact on the thought process that leads to model development as well as on the way a model is presented to the computer (Evans 1989). For this reason, the superiority of one strategy over another has been the subject of much discussion and has led to several comparisons over the years (Hills 1973, Zeigler 1976, Hooper and Reilly 1982, Birtwistle et al. 1985, Hooper 1986). Overall, these investigations have concluded that all strategies are equally general and powerful in terms of being or not being able to represent specific problems. Particular strategies, however, make modeling certain classes of problems easier and this makes them more suitable for certain tasks. In fact, one of the main objectives of this paper is to illustrate that in construction Activity Scanning is indeed a more natural and effective strategy than Process Interaction.

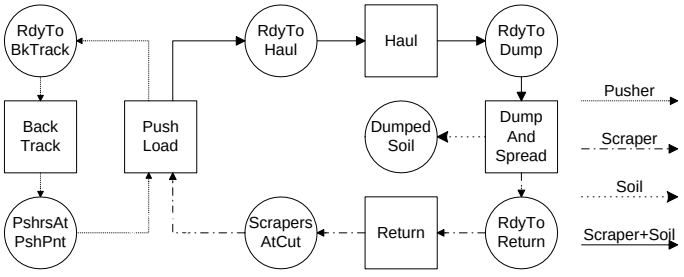

Figure 1 - ACD for scraper and pusher operation

Effect of Simulation Strategy on Model Development

The effect of simulation strategy on model development in construction is best illustrated by using both Process Interaction and Activity Scanning to model the same process. The example chosen for this comparison is a simple earthmoving operation where a pusher and scraper work together to push-load scraped material into the scraper's bowl. The pusher then backtracks to get ready to push-load again and the scraper hauls, dumps and spreads, and returns for another push-loading. A more complex version of this problem will also be used to illustrate why Activity Scanning is the natural approach for modeling complex construction operations in detail.

Activity Scanning Model for a Simple Scraper and Pusher Operation

The first step in developing an AS simulation model is to identify the various activities that can take place, the conditions necessary for these activities to start, and their outcomes when they end. Start-up conditions and outcomes are typically best described in terms of the *resources* involved and their *state* (i.e., their condition). Table 1 summarizes the activities for this operation, the conditions under which they can start, and their outcomes. Activity *PushLoad*, for example, can start when a pusher is waiting at the pushing point and a scraper is waiting at the cut. Its outcomes are a loaded scraper ready to haul and a pusher ready to backtrack.

The conditions necessary for activities to start are often the outcomes of other activities. This makes it easy and natural to represent the information in Table 1 using an activity-oriented network as shown in Figure 1. This type of network is called an Activity Cycle Diagram (ACD) and consists of alternating circles and rectangles connected with links. The rectangles are called *activities* and represent tasks performed by one or more resources. The circles are called *queues* and represent inactive resources in specific states. Each queue in Figure 1 corresponds to an entry in Table 1 under the columns "Conditions Needed to Start" or "Outcome of Activity". Similarly, each activity in Figure 1 corresponds to a row in Table 1. Queues and activities are connected with directional *arcs* that indicate whether the resources in a queue are required to start an activity (from the queue to the activity) or the result of an activity (from the activity to a queue). In practice, an Activity Cycle Diagram can be constructed without the need to create a table first. A

common approach is to draw the activity cycles of the different resources separately and then combine the cycles into a network by joining them where they have activities in common (Halpin 1992, Sczymankiewics et al 1987).

Activity Cycle Diagrams are intuitive representations and their activity-oriented paradigm makes them easy to communicate to persons familiar with other modeling tools (such as CPM precedence networks). ACDs evolved from *wheel-charts* that were used as blueprints for the construction of GSP simulation programs much as flowcharts are used to construct conventional programs (Tocher 1964). ACDs were made popular by the HOCUS simulation system in the late sixties (Hills 1971). The familiar CYCLONE networks used extensively for construction simulation modeling over the past twenty years represent yet a further extension and refinement of ACDs as do the networks used by RESQUE (Chang 1986), COOPS (Liu 1991), CIPROS (Odeh 1992), and STROBOSCOPE (Martinez 1996).

Activity Cycle Diagrams are a natural means for representing Three-Phase Activity Scanning simulation models at the conceptual level and serve as a high-level representation of the main elements of the simulated process. Typically, it is neither necessary nor convenient for an ACD to show more detail as this may obscure the main model structure. An AS model uses the ACD as a blueprint for developing a detailed computer program that repeatedly checks for the conditions necessary for each activity to start. When the conditions are satisfied, the program performs the appropriate actions. These actions typically include acquiring the resources that enabled the activity to start, determining how long the activity will last, holding the acquired resources for the duration of the activity, and releasing the resources when the activity ends.

In some cases, *all* of the conditions needed for an activity to start are provided by the completion of another activity. The only condition for activity *Haul* to start, for example, is the availability of a loaded scraper ready to haul. This condition also happens to be one of the outcomes of activity *PushLoad*. Consequently, the conditions needed for activity *Haul* to start are always and immediately satisfied every time activity *PushLoad* ends. In Three-Phase AS terminology, the start of activity *Haul* is *bound* to the end of activity *PushLoad*. Three-Phase Activity Scanning takes advantage of this observation to speed up program execution by distinguishing between *conditional* activities and *bound* activities. The Three-Phase approach is more efficient than pure Activity Scanning because the simulation program does not have to scan bound activities for startup conditions—they simply start whenever the activity to which they are bound ends. In the ACD of Figure 1 all activities are bound except for *PushLoad* which is conditional.

In addition to representing activities, an Activity Scanning simulation model must also model the resources involved in the process, such as materials, labor and equipment. Resources and their state are equally important because they constitute the predominant requirement for activities to take place and because the primary effect of activities is to change their state. In AS all resources are considered equal. No distinction is made between those that serve and those being served or between materials, labor and equipment. In the pusher and scraper operation, for example, except for the fact that each represents a different resource, pushers, scrapers and soil are all treated alike. As described below, this equitable modeling approach is in direct contrast to that taken by Process Interaction and is one of the main reasons why Activity Scanning is more natural for modeling construction operations.

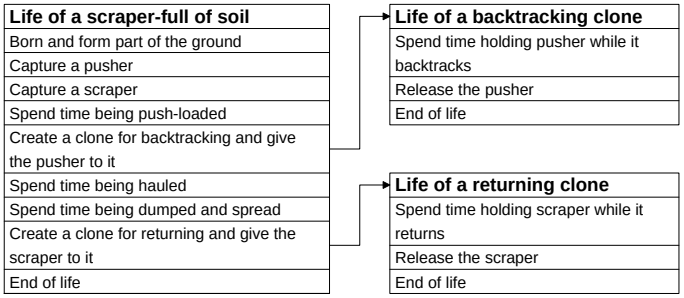

Figure 2 - Moving entities in a PI model of the scraper and pusher operation

Process Interaction Model for a Simple Scraper and Pusher Operation

The first step in developing a Process Interaction simulation model is to identify the entities that flow through the system and the scarce resources for which these entities compete. This is particularly important because, typically, moving entities cannot compete for the acquisition of other moving entities, and scarce resources (being completely passive) cannot compete for other resources. Moreover, a Process Interaction model must adopt the point of view of the moving entities and this makes certain choices much more convenient than others. Thus, more than anything else, the prudent choice of moving entities and scarce resources determines the ease of model development as well as the effectiveness of the resulting simulation model. This choice, however, is by no means obvious and as shown below it often makes modeling the dynamic interactions that occur in construction quite challenging.

In the scraper and pusher operation, for example, it is possible to think of the scrapers, the pushers, or the soil as moving entities. It is also possible to think of the scrapers or the pushers as scarce resources. Figure 2 illustrates a Process Interaction model where the soil is the main moving entity and the scrapers and pushers are scarce resources. Notice that because process interaction must be triggered by moving entities, it is necessary to clone (split) each unit of soil (a moving entity) to model the return of a scraper and pusher back to the loading area correctly.

At first glance, the PI model shown in Figure 2 may appear to be appropriate. This would be the case if units of soil arrive to the system at some rate, wait for processing, and then leave (which is what typically happens in a manufacturing scenario). In the scraper and pusher operation, however, all soil units exist simultaneously from the very beginning of the process. Thus, a computer program implementing this PI model would have to spend a significant amount of time constantly checking to see if each separate unit of soil (the moving entity) can advance to the next step in its life. Furthermore, modeling and storing each scraper-full of soil as a separate entity would waste a significant amount of computer memory without providing any benefits.

A different PI model is shown by the flowchart in Figure 3 where scrapers are the main moving entities and pushers are the scarce resources (in this case soil is modeled as a secondary moving entity). This flowchart is in GPSS notation (Schriber 1990) because there is no standard way of representing a PI model. Flowcharts using other notations (e.g., SIMAN) or PI-

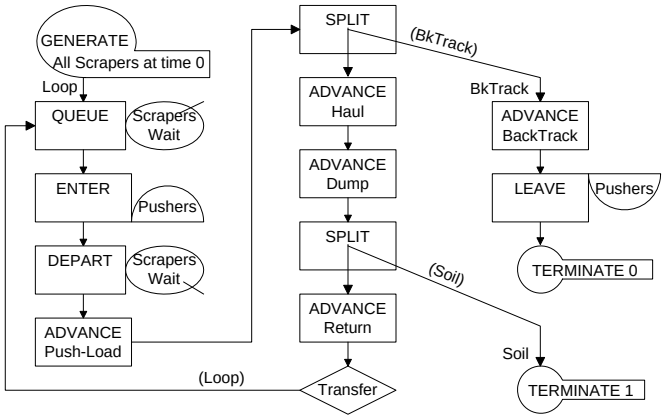

Figure 3 - Process Interaction flow chart for scraper and pusher operation

oriented networks (e.g., SLAM) use slightly different shapes and names for the blocks but are conceptually similar. For this earthmoving example, the processes followed by scrapers and pushers are cyclic, and therefore it is necessary to create clones regardless of which resources are chosen as moving entities. In this case, a scraper clone is needed to occupy each pusher and keep it busy while it returns and gets ready to push-load again. Modeling scrapers as moving entities as shown in Figure 3 may in fact be the best possible choice for this model because scrapers undergo a much more detailed process when compared to pushers.

Each node in this flowchart is called a block. The GENERATE block creates all the scrapers to be used in the operation at the beginning of the simulation. A scraper immediately joins a queue (QUEUE) where it waits to capture a pusher (ENTER). After successfully capturing a pusher, the scraper leaves the queue (DEPART) and spends some time being push-loaded (ADVANCE). The scraper then splits (SPLIT) into two entities. The original entity spends some time hauling (ADVANCE) and subsequently dumping (ADVANCE). The copy is necessary to retain the pusher captured until backtracking is done. After backtracking, the scraper copy releases the pusher (LEAVE) and self-destructs (TERMINATE). After dumping, the original scraper splits again (SPLIT). The original entity spends some time returning (ADVANCE) and goes back (TRANSFER) to rejoin the original queue (QUEUE). The second copy of the scraper may be thought of as a scraper-full of soil. It simply self-destructs (TERMINATE) and signals that one unit of production is complete.

These examples illustrate how important it is to select the appropriate resources to model as moving entities and as scarce resources. For systems where many types of resources interact and have multiple functions, this choice may not be immediately obvious or in any way trivial, as shown below.

Modeling Construction Operations—Process Interaction Vs. Activity Scanning

Although any strategy can be used to model any problem, particular strategies make modeling certain classes of problems easier. The Process Interaction strategy is well suited for most manufacturing applications where materials arrive to a system and undergo a rather fixed processing pattern before they leave as finished products. These systems can potentially include several types of resources (servers), but the resources

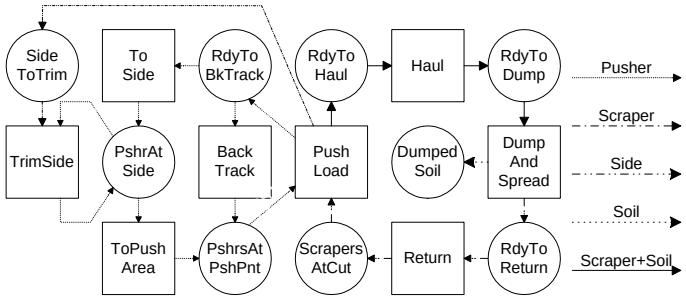

Figure 4 - ACD for scraper and pusher operation with side trimming

themselves are rarely in more than a few states. Furthermore, they tend to work alone or in collaboration with a very small number of other resources.

In contrast, Activity Scanning is well suited for systems that include many interacting resources that can be in numerous states and where logical complexities are best described in terms of the conditions required to carry out activities. Most construction operations are of the latter type and are therefore more easily represented using the AS paradigm.

The advantages of Activity Scanning vs. Process Interaction are best illustrated with an example such as the one shown by the ACD of Figure 4. In this extended version of the scraper and pusher operation, it is necessary to periodically trim the sides of the cut area. Thus, push-loading now generates "side in need of trimming". Activity *ToSide* represents the pusher moving towards the side, activity *TrimSide* represents the actual side trimming, and activity *ToPushArea* represents the pusher moving to the pushing point.

Activities *ToSide* and *BackTrack* compete for a pusher ready to backtrack. Which of the two takes place depends on their relative priorities, which in turn depend on the current conditions at the cut. Activity *ToSide* may be of higher priority when the amount of side in need of trimming is excessive. Activity *BackTrack* may have higher priority when several scrapers are idle waiting to be push-loaded. (For simplicity we are assuming that the excavation material is hard and does not allow scrapers to self-load or push-pull, thus making push-loading necessary.) When there is side in need of trimming, activities *ToPushArea* and *TrimSide* compete for a pusher at the side. In both cases, the details for determining the conditions under which one activity has priority over another are not shown in the ACD.

For the revised version of the earthmoving operation shown in Figure 4 idle pushers can be in three different states: at the pushing point, ready to backtrack, or at the side. In the previous version shown in Figure 1 the ready-to-backtrack state could be disregarded because activity *Backtrack* was bound to activity *PushLoad*. This is no longer the case and the number of meaningful non-active states for pushers has increased from one to three.

Attempting to represent the new problem with a Process Interaction model is significantly more difficult. A scraper must now capture a pusher when the latter is idle *at the pushing point*. The pusher, however, could be idle *at the side* or idle and *ready to backtrack* and thus not be suitable for capturing by the scraper. It suddenly becomes more convenient to switch the point of view of the entire Process Interaction model and treat pushers as moving entities and scrapers as scarce resources.

Modeling pushers as moving entities makes it necessary to resolve how to route a moving pusher after it push-loads, or

when it is at the side, while at the same time managing scrapers as scarce resources. A pusher must capture a scraper before push-loading but must not relinquish it when done. Instead, the pusher must split and create a clone that will haul, dump, return, and then relinquish the scraper. Furthermore, several other resource interaction issues must be resolved before a valid PI model can be created.

Although this example is small, it does illustrate the difficulties associated with using Process Interaction to model even simple construction processes (the complete PI model for this example is not shown here due to space limitations). In particular, switching the roles of pushers and scrapers as moving entities and scarce resources requires a shift in modeling paradigm that is far from obvious. As a construction operation becomes more detailed and the number of interacting resources increases, the convenience of simulation modeling using Process Interaction decreases and the advantages of Activity Scanning become more significant.

It is common in construction for several resources to collaborate to accomplish a task. In most cases, an activity cannot start unless all these resources are available and in the correct state. If any of the resources is not available or in the proper condition then the task cannot be carried out. Situations such as these are naturally and automatically represented by ACDs but pose difficulties for PI tools. As an example, consider the PI model of Figure 2 where the main moving entity is a scraper-full of soil. The soil first tries to capture a pusher and then a scraper. If a pusher is available but a scraper is not, the soil captures a pusher and then waits until a scraper becomes available. During this waiting time the pusher is committed and cannot do anything else. That is, it cannot be captured by a moving entity in another process where a pusher does not need to collaborate with a scraper (such as when "a side in need of trimming" tries to capture it to trim the sides). D.W. Halpin describes this type of problem in detail in his dissertation (1973).

Although some of the newer PI tools (e.g., ProModel) have features that enable all-or-none resource acquisition, they still force the differentiation between moving entities and scarce resources. When many resources interact, one must be treated as a moving entity and all the others as scarce resources. Dealing with the multiple states of all these resources decreases the convenience of using a PI tool geometrically.

Having many interacting resources in a Process Interaction model can also lead to the problems of entanglement and deadlock (Tocher 1979). Entanglement is a complex problem that occurs when the processes followed by several moving entities of different types get intertwined. Deadlock occurs when a moving entity captures one scarce resource (e.g., a scraper) and is waiting to capture a second resource (e.g., a pusher), while another entity has captured the resource needed by the first entity (i.e., the pusher) and is waiting to capture the resource in possession of the first entity (i.e., the scraper). Neither entity can continue its path because each has the resource that the other needs.

This brief comparison of simulation strategies indicates that Activity Scanning makes it much easier to model complex processes with many interacting resources. The primary reasons for this can be summarized as follows:

- The fundamental point of view of a model does not shift as the number of resources or the complexity of the problem increases.
- There is no need to differentiate between scarce resources and moving entities.
- The all-or-nothing requirement for resource acquisition (which is common in construction) is satisfied automatically.
- It is not necessary to be concerned about entanglement or deadlock because they cannot occur. (When modeling complex construction processes using PI tools these must be explicitly avoided.)
- In construction the focus is on the tasks that need to be performed and their requirements and not on what each resource needs to do next.

Simulation System Flexibility and Programmability

In a general sense, the flexibility of a general-purpose simulation system is independent of the strategy it adopts, whether it is Process Interaction or Activity Scanning. Programmability, however, significantly increases flexibility.

It is possible to provide simulation systems with a certain level of flexibility by building capabilities into the tool. Once a capability is built-in, it can be used within the limitations of its design. Consider, for example, a discrete-event simulation tool with the capability to model learning-curve effects by adjusting the parameters of an activity's duration as a function of the number of times the activity has been performed. For instance, the parameter for the duration of the n^{th} activity occurrence, P_n, could be a function of the parameter for the duration of the first occurrence, P_1, and a constant s, $P_n = P_1 * n^s$. Such a tool may have a facility or a template where the user can indicate that learning effects should be considered and where the values of P_1 and s are specified. The addition of this template gives the tool the flexibility to model learning-curve effects but at the same time limits its capabilities to this particular learning model. Furthermore, the addition of this template increases the complexity of the tool because more has to be learned before it can be used.

Instead of continuously adding template-based capabilities to a simulation system, it is possible to attain much more flexibility by providing the system with end-user programmability. (This obvious strategy has been currently adopted by many software products, especially those that support a version of the BASIC language). As an example, consider the case of a simulation tool that supports the simplest level of programmability, the ability to use a formula to define the parameter for the duration of an activity. It is then possible to simply write the appropriate formula as part of the model and achieve the same learning-curve effect described above. For example, consider a steel erection activity named *Erect* for which P_1 = 11.8 hours and s = -0.129. The parameter for the duration of the activity could then be specified as 11.8**Erect.TotInst*^-0.129, where *Erect.TotInst* is a variable that dynamically returns the number of times that the *Erect* activity has been performed. Thus, simple programming can be used to represent not just this but any other learning-curve model as well. Moreover, it can be used to model any number of other situations that require the use of non-stationary dynamic parameters. Flexibility is limited only by the complexity of the formula allowed by the language and by the comprehensiveness of the dynamic data that is made accessible. It must be pointed out, however, that this increase in flexibility is accompanied by an increase in system complexity—it is now

necessary to know the programming syntax that can be used to prepare a formula and to access dynamic data.

It is obvious that programmable simulation systems and simulation programming languages can be very powerful but at the same time can be more complex and thus require a stronger commitment for their mastery. Simple systems can be learned very easily but are limited to the capabilities that are built into the tool. Certain object-oriented concepts can be applied to the design of systems with the purpose of hiding complexity to the novice without removing the programming capabilities that give flexibility to the advanced user. It is very difficult, however, to provide flexibility and extreme simplicity in the same tool. The degree to which this is achieved is a measure of how good the system design is and determines the long-term success of the simulation program in practice.

Simulation System Features

Features are yet another dimension of simulation systems that affect convenience and ease of use. Simulation system features do not depend on the issues described above, such as programmability (flexibility) or simulation modeling strategy (the suitability for modeling certain types of systems). Instead, features include capabilities such as graphical user interfaces, interactive debugging and tracing features, production of presentation quality reports with tables and charts, and animation. In the case of a general-purpose or simulation programming language, features are a characteristic of the specific implementation and not of the language itself. Different vendors of the same ANSI C++ (Stroustrup 1998) language, for example, offer different sets of features in their compilers. Although such features may not be part of the underlying language, they are an integral part of the system and determine its acceptance and long-term use in practice. Thus, construction simulation tools should not only be based on activity scanning and provide for programmability, but must also incorporate a rich set of powerful features, such as a graphical interface, a powerful model debugger, and animation.

GENERAL-PURPOSE CONSTRUCTION SIMULATION TOOLS

ACDs are overwhelmingly convenient for construction operations modeling. In practice, however, simulation models must be implemented using actual systems with specific flexibility and feature sets. For many years, the available ACD-based simulation systems had been limited to non-programmable tools that were adequate and extensively used for instructional purposes and for modeling systems that were not too complex or detailed. Advanced programmable AS systems were developed primarily in England during the sixties and seventies and were proprietary and practically unknown in the United States. Perhaps the best example is HOCUS, a system with significant potential for modeling cyclic construction operations (Halpin 1973).

In contrast, advanced and feature-rich PI simulation languages for modeling manufacturing and service-oriented systems have been widely available since the sixties. These PI tools have also been used to model various relatively complex construction operations despite the significant effort required to develop appropriate simulation models (e.g., Ashley 1980). In the past, these choices were necessary because complex construction processes require a flexibility that was not available in ACD-based tools. Thus, it became common for construction researchers to conceptualize, reason, and explain complex models using ACDs, but to implement them using a PI-based language. A recent example is Resource-Based Modeling (RBM) where ACDs were used to explain the concept and SLAM II was used for the implementation (Shi and AbouRizk 1997). The authors explained that the need to convert the CYCLONE model [an ACD representation] to an equivalent SLAM II simulation model [using a PI based language] was necessitated by issues of "flexibility".

Fortunately, this no longer has to be the case. In the remainder of this section we will examine the development of construction simulation tools and conclude that we have at last reached the point where the variety of available AS-based simulation tools for construction is as broad as that for other industries.

Petri Nets

A class of AS-based discrete-event simulation modeling systems are founded on Petri Net concepts. A Petri Net is a formal graphical representation for the analysis of systems with concurrency (Petri 1966). Petri Nets are supported by a general theory of discrete parallel systems that generalizes the theory of automata in a way that allows the expression of concurrently occurring events. Petri Net theory provides formal methods of analysis to determine whether a specific configuration of a system can be reached from another configuration, whether a system is reversible, whether a complex system can be simplified, whether two different systems are functionally equivalent, and many other similar analyses (Jensen 1992).

Since the late eighties, Petri Nets have also been used for discrete-event simulation. The nodes in a Petri Net alternate between *transitions* and *places*. Transitions are typically shown as bars, but are also often shown as squares or rectangles. Places are shown as circles. These nodes are connected with directional *edges* (links). *Tokens* initially reside in places. In the basic case, a transition *fires* when each place preceding it contains at least one token. Upon firing, the transition removes tokens from preceding places and deposits tokens in succeeding places. There is often a *holding period* that indicates a delay between the time of transition firing (removal of tokens from preceding places) and the time at which tokens are deposited in succeeding places. It must be pointed out, however, that the use of a holding period to support the passage of time robs the methodology of its power because it is incompatible with most formal Petri Net analyses (Jensen 1992).

From this discussion, it is clear that Petri Nets are practically identical to ACDs in functionality and appearance when used to perform discrete-event simulation: a place is analogous to a queue, a transition to an activity, an edge to an arc, a token to a resource, and the holding period of a transition to the duration of an activity. Petri Nets are in fact used to teach Three-Phase AS concepts in computer science programs (Schruben 1996). In addition, several papers on the relationships between Petri Nets and Three-Phase AS have been published (e.g., Lin and Lee 1993).

Given that for all practical purposes Petri Nets are indeed ACD tools, they show potential for use in the simulation of construction operations. Recent literature already shows uses of Petri Nets for construction modeling (Damrianant and Wakefield 1997, Sawhney 1997). Numerous Petri Net-based

simulation tools exist with different capabilities and features. When used to model construction operations, however, Petri Nets exhibit certain important limitations. One of them is the inability to represent directly bulk materials, such as soil or aggregate, that may exist or be transferred in measurable continuous quantities. Thus, resource quantities expressed as real numbers (i.e., non-unitary) must be modeled by manipulating data attached to discrete tokens. Another limitation is that the rules for configuring Petri Nets are strictly the same as those for pure ACDs and do not allow for the explicit identification of bound transitions (activities) and the elimination of the corresponding superfluous places (queues). These extensions to ACD's have been adopted by simulation tools designed specifically for construction and have proven to be very convenient. Lastly, the language and formal nature of Petri Nets give them a complexity that is not matched with a corresponding increase in flexibility for those interested only in discrete-event simulation.

Systems Designed Specifically for Construction

Several general-purpose simulation systems have been designed specifically for construction operations. They are based on ACDs with extensions that make them more suitable for modeling construction operations. Although several systems are briefly mentioned here, the emphasis is on CYCLONE (Halpin & Riggs 1992) and STROBOSCOPE (Martinez 1996) because they exhibit the least overlap in the flexibility/simplicity spectrum that characterizes general-purpose simulation tools.

CYCLONE

CYCLONE (Halpin and Woodhead 1976) is the oldest and most used general-purpose simulation tool designed specifically for construction. Although in his dissertation Halpin (1973) described the CYCLONE logic in terms of GERT networks (Pritsker 1966), the flowcharts that describe the CYCLONE simulation logic in (Halpin and Riggs 1992) support the notion that CYCLONE is an Activity Scanner where the ACD is the simulation model itself and not just a blueprint for a simulation program. There have been at least four different implementations of CYCLONE: main-frame CYCLONE (Halpin 1976), Insight (Kalk 1980), UM-CYCLONE (Ioannou 1989), and Micro-CYCLONE (Halpin 1990). Numerous capabilities and features have been built into the various implementations of CYCLONE (e.g., Dabbas and Halpin 1982, Bernold and Halpin 1985, AbouRizk and Halpin 1990, Lutz et. al. 1994, Huang and Halpin 1994). CYCLONE has also been used to model many construction operations including concrete batch plants (Lluch and Halpin 1982) and tunneling (Touran and Asai 1987).

CYCLONE ACDs enhance pure ACDs with two different types of extensions: *conceptual* and *functional*. **Conceptual** extensions allow for the explicit classification of activities as *conditional* (called COMBI and shown by squares with a slash on the top-left corner) or *bound* (called NORMAL and shown as simple squares) and for the elimination of superfluous queues. These extensions make conceptual CYCLONE ACD's more compact, appropriate and appealing for construction operations. Consider the conceptual CYCLONE ACD for the simple scraper and pusher operation shown in Figure 5. It uses four queues and four arcs less than the pure ACD of Figure 1. In addition, it makes sequences of chained activities explicit. It is, for example, quite clear that the activities *PushLoad*, *Haul*, *DumpAndSpread*, and *Return* follow each other.

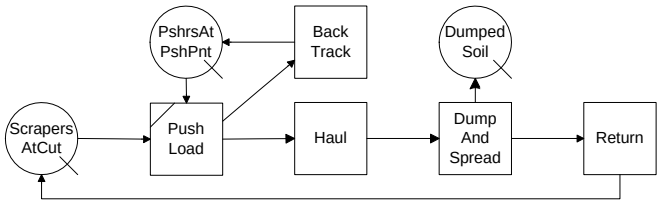

Figure 5 - CYCLONE Conceptual ACD for simple scraper and pusher operation

Functional extensions enable the CYCLONE ACD to be a complete and unambiguous representation of a simulation model. These extensions include additional functions and nodes (GENERATE, CONSOLIDATE, COUNTER), probabilistic branching for activities followed by more than one other activity, the assumption that all resource flows and requirements are unitary, and the assumption that all resource units are indistinguishable and interchangeable. With these enhancements and assumptions it is possible to create a functional CYCLONE ACD that represents a simulation model with all its details.

To illustrate the power and specificity of CYCLONE's ACD representation consider the scraper and pusher operation with side trimming, shown conceptually in the ACD of Figure 4, with the following specific details:

1) Each scraper has a capacity of 20 cubic meters.
2) Every 20 push-loads generate the need for one side-trimming pass.
3) After push-loading a scraper, the pusher moves towards the side to trim if there are at least 5 passes worth of trimming to do.
4) The pusher remains trimming at the side until all required trimming passes are complete. The pusher then moves to the push-loading area.

The operation can be entirely represented with the functional CYCLONE ACD of Figure 6. All nodes in this network are numbered to provide a way to refer to them in a shorthand manner. The actual node numbers are meaningful only for COMBI activities – those with a lower number have priority over those with a higher number. Nodes 3, 10, and 13 have been added to what would otherwise be a conceptual ACD. Node 3 is a consolidator that holds the resources released by activity *PushLoad* until 100 are accumulated and then releases one

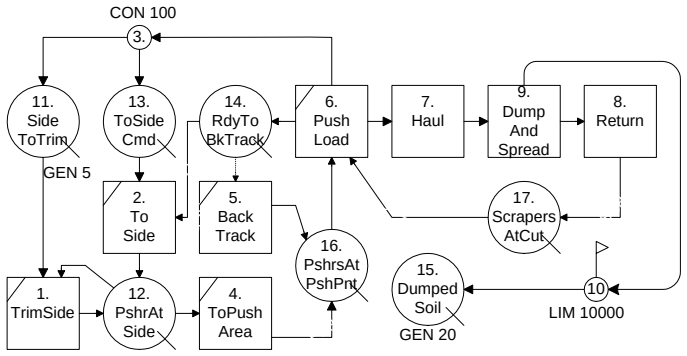

Figure 6 - CYCLONE functional ACD for pusher and scraper operation with side trimming

resource to queue 13 and another resource to queue 11. The GEN 5 for queue 11, *SideToTrim*, multiplies the incoming resource by 5 to allow activity *TrimSide* to take place 5 times (the GEN function generates 5 resources for each one coming in). Since activity *ToSide* has priority over activity *BackTrack*, the presence of a resource in queue 13 allows activity *ToSide* to take hold of the resource in queue *RdyToBkTrack* before activity *BackTrack* does. Similarly, the higher priority of activity *TrimSide* over activity *ToPushArea* keeps the pusher trimming the sides before moving it to the push-loading area. The GEN 20 at queue 15 multiplies each resource by 20 to convert scraper-loads to cubic meters of soil. The limit of 10,000 on node 10 indicates that the simulation should run until 10,000 scraper-loads have been dumped.

Missing from Figure 6 are the durations of the different activities, the number of initial resources in each queue, and the maximum time limit for the simulation. These details have been omitted for clarity but could have been included to make the network drawing and its annotations a complete representation of the model. For example, the duration of an activity could be defined by specifying a probability distribution and the numerical values of its parameters.

The fact that a CYCLONE model can be completely described by a functional ACD is responsible for both its advantages and limitations. On the positive side, it keeps the modeling methodology simple and easy to learn. It also facilitates communication because everything is contained in the network. The limitations are that many simplifying assumptions must be made when modeling operations that include complex logic, multiple resources with different properties, or non-stationary parameters based on models that have not been built into the CYCLONE implementation. Some examples of these simplifying assumptions and their consequences as well as a comparison between CYCLONE and SLAM II can be found in (Gonzalez-Quevedo et al 1993) and (Gonzalez-Quevedo 1995). CYCLONE and SLAM II are two simulation tools that differ in both simulation strategy and programmability. As expected, the authors concluded that CYCLONE is simpler and more natural for construction operations and that SLAM II is more powerful. The simplicity and natural appeal for construction found in CYCLONE is primarily due to the use of ACDs as a means of representation. The power and flexibility found in SLAM II is primarily due to the fact that it is a mature simulation programming language.

CYCLONE has inspired the development of other simulation systems (such as those described below) and its continued widespread use has helped make construction simulation modeling popular.

RESQUE

RESQUE (Chang 1986) incorporates CYCLONE's conceptual and functional extensions, but the model is not limited to the information conveyed by the network. A RESQUE model also includes an *overlay* that defines resource distinctions and increases simulation control. The overlay provides it with significant flexibility insofar as recognizing distinctions among resources that flow through the same path. For example, the time to load a truck can be sensitive to its type (size) and an activity may be prevented from starting unless the contents of a queue exceed a particular amount.

COOPS

COOPS (Liu 1991) is an object-oriented system that enhances CYCLONE's conceptual and functional extensions with some relaxed node precedence rules. COOPS models are defined via a graphical user interface where all resources are treated as individually identifiable objects to provide statistics from the point of view of each individual resource. These are in addition to the traditional statistics that reflect the point of view of the activities performed or the queues visited. In addition, COOPS allows for the generation and consolidation of resources at links and uses calendars that can be used to preempt activities during work breaks.

CIPROS

CIPROS (Odeh 1992) is both a process level and project level planning tool. It contains an expandable knowledge base of construction techniques and methods, and makes ample use of a hierarchical object-oriented representation for resources and their properties. The resource characterization capabilities in CIPROS go beyond those in RESQUE to allow multiple real properties for resources as well as more complex resource selection schemes. CIPROS also integrates process-level and project-level planning by representing activities through process networks, all of which can use a common resource pool.

STEPS

STEPS (McCahill and Bernold 1993) is a general-purpose system based on pure ACDs that was developed for the U.S. Navy. STEPS supports the notion of different resource sizes in the same queue.

STROBOSCOPE

STROBOSCOPE is a general-purpose simulation programming language with direct support for Three-Phase Activity Scanning and Activity Cycle Diagrams. STROBOSCOPE ACDs represent simulation models only at the conceptual level—the details of a model are specified with other mechanisms that rely on programmability. At the *conceptual* level, the elements used in a STROBOSCOPE ACD are a superset of those in CYCLONE (for example, STROBOSCOPE allows for the explicit identification of bound activities with the elimination of the corresponding superfluous queues). In addition, STROBOSCOPE introduces five new nodes and four special types of links of *conceptual* significance. STROBOSCOPE models, however, do not rely on *functional* CYCLONE elements (e.g., Generate, Consolidate, Counter) and are not subject to any of the simplifying assumptions found in *functional* CYCLONE models (for example, resources of the same type can be distinguished from one another and each can have individual properties). Each modeling element in a STROBOSCOPE ACD has methods (attributes) that define the element’s behavior. Each method (attribute) has a default implementation that reflects the element’s most common behavior but which can be redefined to achieve the desired effect. Thus, the functional details of a model are mostly specified by re-defining these methods. The code used to redefine a method can be as simple as a number, but can also consist of complex expressions that call functions, that access

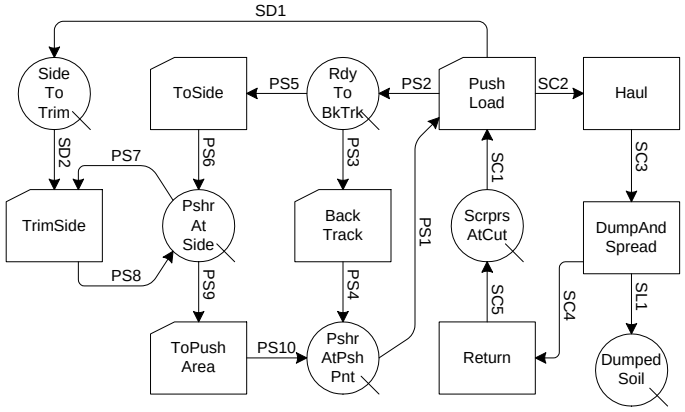

Figure 7 - STROBOSCOPE ACD for pusher and scraper operation with side trimming

the dynamic state of the model, and that manipulate arrays, statistics collection objects, or files.

To illustrate the development of a STROBOSCOPE simulation model consider the pusher and scraper operation with side trimming described conceptually by the ACD of Figure 4 and shown as a STROBOSCOPE ACD in Figure 7. The two main differences between these representations are the elimination of superfluous queues (and the explicit identification of bound activities) and the use of names to identify links in the STROBOSCOPE version. Link names are used to redefine link-related methods and to access link-related dynamic data at runtime. By convention, the names used for links are composed of a two-letter abbreviation of the type of resource that flows through and of a number that gives an idea of flow sequence. In this case, the *PS* links are for pushers, the *SC* links for scrapers, the *SD* links for side, and the *SL* link for soil.

In order to define a functional simulation model for this operation it is also necessary to establish priority rules for the activities *ToSide, BackTrack, TrimSide* and *ToPushArea*. For an earthmoving operation with the same specific details as those used to build the functional CYCLONE ACD of Figure 6, it is necessary to redefine several modeling element methods as shown in Table 2. Except for activity durations, which for brevity are omitted here, it is not necessary to redefine any other element method because the default behavior is exactly what is needed. These redefinitions translate into plain English as follows:

- After push-loading create one unit of "side" for trimming. (In the absence of this statement zero units of "side" would be created since none are acquired for push-loading.)
- The priority of activity *ToSide* depends on the number of "side" units in need of trimming (*SideToTrim.CurCount*). This priority becomes increasingly greater than normal (i.e., greater than zero) as the number increases past 100, but is below normal when the number is less than 100. (By default the priority of all activities is zero; this "tie" is broken by the order in which the activities are defined.)
- The priority of activity *TrimSide* is always greater than normal.
- Activity *TrimSide* consumes 20 "side" units every time it takes place. (By default, only 1 unit of "side" would be consumed.)
- After dumping and spreading, create 20 units of soil. (In the absence of this statement activity *DumpAndSpread* would release zero units of soil since it did not receive any.)

Table 2 - STROBOSCOPE method redefinitions for a simple scraper and pusher operation with side trimming

Element	Method	Redefinition
SD1	Release Amount	1
ToSide	Priority	SideToTrim.CurCount-100
TrimSide	Priority	10
SD2	Draw Amount	20
SL1	Release Amount	20

Although used for a simple means in this case, the dynamic and intelligent priority of the *ToSide* activity gives a glimpse of the flexibility that comes with programmability. This flexibility, however, comes at a price. In order to use the system effectively it is necessary to know that appending "*.CurCount*" to the name of a queue creates a dynamic variable that returns the queue's current content at runtime. It is also necessary to understand how a model will behave by default, and how to alter that default behavior to achieve the desired results.

The necessary redefinitions for a slightly more complex and detailed variation of the scraper and pusher operation, based on the same STROBOSCOPE ACD, are shown in Table 3. The functional details specified in this table for the more complex example are:

- The amount of "side" in need of trimming generated by a push-load is not constant but rather follows a Normal probability distribution.
- The amount of "side" that is trimmed in one pass is also not constant and follows another Normal probability distribution.
- Backtracking (*BackTrack*) and returning the pusher from the side to the push area (*ToPushArea*) become high priority activities if the number of scrapers at the cut is high (i.e., greater than 3 and 4 respectively).
- The duration of activity *BackTrack* is a linear function of the time it took to push-load. The previous push loading time is the difference between the current time (*SimTime*)

Table 3 - STROBOSCOPE method redefinitions for a more detailed and complex scraper and pusher operation with side trimming

Element	Method	Redefinition
SD1	Release Amount	Normal[1,0.05]
SD2	Draw Amount	Normal[20,1.5]
ToPushArea	Priority	ScrprsAtCut.CurCount>4 ? 20 : 0
BackTrack	Priority	ScrprsAtCut.CurCount>3 ? 20 : 0
BackTrack	Duration	0.25+0.4*(SimTime-PushLoad.LastStart)
Haul	Duration	(800+2*PushLoad.TotInst)/Pert[25,40,45]/16.7

and the time at which the last push load started (*PushLoad.LastStart*).

- The duration of activity *Haul* is based on an initial distance of 800 meters that is incremented by 2 meters after each scraper trip (*PushLoad.TotInst* returns the total number of times that *PushLoad* has taken place). It also depends on the scraper's average speed (in km/hr) as given by a Pert distribution with P_0=25, Mode=40, and P_{100}=45.

These example redefinitions were selected to illustrate some of the issues that are easily handled through programmability but pose significant difficulties otherwise. They include probabilistic resource production and consumption; dynamic activity priorities based on the state of the system; dependence of the duration of one activity on the duration of another, or on any other dynamic system characteristic (such as the constantly changing haul distance); and the capability to embed a sampled random variate in a more complex expression. The inclusion of each one of these capabilities in a non-programmable system is a non-trivial undertaking that requires significant independent research.

For the sake of brevity, issues relating to programmability were illustrated in a context that exists in every general-purpose simulation system based on ACDs (e.g., priorities and durations). STROBOSCOPE makes available many other redefinable methods that allow tailoring the behavior of modeling elements very precisely. It also includes programmable modeling concepts that support detailed and complex operations. These include bulk resources; uniquely identifiable discrete resources that can carry information (characterized resources); organization of resources in containment hierarchies of unlimited depth (compound resources); intelligent and dynamic resource routing (forks and dynaforks); non ACD-derived control of activity startups (semaphores); and advanced experimental control and random number stream management for the implementation of variance reduction techniques, sensitivity analysis, and automated replications.

The STROBOSCOPE language also provides facilities for writing code to manipulate variables, arrays, statistics collection objects, files, and the like, at the occurrence of specific events attached to modeling elements during simulation (e.g., write formatted output to a disk file every time a scraper flows through link SC2). Programming capabilities of another nature include the ability to extend the language with dynamic link libraries written in high-level programming languages such as C++, and the ability to integrate STROBOSCOPE as a component in a multi-tool decision support system by using its OLE Automation interface.

As with any programming language, it is impossible to mention all aspects of STROBOSCOPE within the limits of this short description. Table 4, however, provides some general data about the language.

STROBOSCOPE has been used to model tunneling operations with variance reduction techniques (Ioannou and Martinez 1996), lean construction processes (Tommelein 1998), paving operations (Harmelink and Bernal 1998), optimum load-growth effects and variable haul distances in pusher and scraper operations (Martinez, Ioannou and Carr 1994), quarry operations (Martinez and Ioannou 1995), impacts of changes in construction work (Cor 1998), and concrete block manufacturing (Sangarayakul 1998) among others. In addition, STROBOSCOPE has been used to create animations of construction operations using PROOF (Ioannou and Martinez 1996a, 1996c, 1996d), to build project-level modeling tools (Wang 1996, Martinez and Ioannou 1997) and special-purpose simulators (Martinez 1997, 1998).

CONCLUSION

The essential characteristics of discrete-event simulation systems are their application breadth, modeling paradigm and flexibility. Application breadth determines whether a system is designed for general-purposes or for the modeling of operations within a specific and narrow domain. The modeling paradigm (simulation strategy) used by a general-purpose tool gives it its flavor and makes it naturally suitable for modeling certain types of systems. The main strategies in use today are Process Interaction (PI), Activity Scanning (AS), and Event Scheduling (ES). Event Scheduling is often used to enhance PI and AS. The combination of AS and ES is typically called the Three-Phase approach. Although systems based on any strategy can be used to model construction operations, it is most convenient to represent them using Activity Cycle Diagrams, which are networks that naturally describe Three-Phase AS models. The flexibility of a tool is independent of its strategy and application breadth. In general, simple tools are very easy to learn and use but are limited in their modeling capabilities. More advanced tools typically involve programming and thus require a stronger commitment for their mastery but are capable of modeling almost any system in detail regardless of its complexity.

Several general-purpose discrete-event systems have been designed specifically for modeling construction operations. These systems are all based on some form of Activity Cycle Diagrams and use the AS or three-phase AS approach. Two of these systems, CYCLONE and STROBOSCOPE, are

Table 4 - STROBOSCOPE language facts

Fact	Number
Conceptual ACD elements (queue, combi, normal, link, dynafork, assembler, etc.)	13
Element methods that can be redefined (duration, priority, strength, draw order, draw when, etc.)	19
Types of dynamically accessible modeling element data (SimTime, queue.*AveWait*, activity.*TotInst*, etc.)	111
Events for performing actions at simulation runtime (On Start, Before End, On Release, etc.)	10
Types of statistics collection objects (collectors, weighted collectors, time-weighted collectors, etc.)	10
Probability distributions (Rnd[], Normal[m,s], Beta[a,b], Gamma[a,b], Pert[a,M,b], etc.)	11
Functions (Abs[val], Ln[val], Sin[val], Confidence[sd,lvl,n], etc.)	134
Statements (CHARTYPE, SIMULATEUNTIL, WHILE, REPORT, etc.)	86
Operators (! - ^ / * - + >= > <= < != == & | ?: etc.)	20
Random number streams	23,000

representative of the range of modeling tools that are now available for use in the analysis of construction operations. CYCLONE is well-established, widely used and simple system that is easy to learn and effective for modeling many simple construction operations. STROBOSCOPE is a programmable and extensible simulation system designed for modeling complex construction operations in detail and for the development of special-purpose simulation tools. These systems indicate clearly that Three-Phase Activity Scanning is the wave of the future for construction simulation.

ACKNOWLEDGEMENTS

The authors wish to thank the National Science Foundation (Grants CMS-9415105 and CMS-9733267) for supporting portions of the work presented here. Any opinions, findings, and conclusions or recommendations expressed in this paper are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

AbouRizk, S. M., and Halpin, D. W. (1990). "Probabilistic Simulation Studies for Repetitive Construction Processes", *Journal of Construction Engineering and Management*, ASCE, 116(4). 575-594.

Ashley, D. (1980). "Simulation of Repetitive-unit Construction", *Journal of the Construction Division*, ASCE, 106(CO2). 185-194.

Balci, O. (1988). "The Implementation of Four Conceptual Frameworks for Simulation Modeling in High-level Languages", *Proceedings of the 1988 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 287-295.

Bernold, L. E. (1989). "Simulation of Nonsteady Construction Processes", *Journal of Construction Engineering and Management*, ASCE, 115(2). 163-178.

Birtwistle, G., Lumow, G., Unger, B., Lucker, P. (1985). "Process Style Packages for Discrete Event Modeling: Experience from the Transaction, Activity and Event Approaches", *Transactions of the Society for Computer Simulation* 2:1 27-56.

Buxton, J. N., and Laski, J. G. (1962). "Control and Simulation Language", *The Computer Journal*, British Computer Society, 5:194-199.

Chang, D. Y. (1986). "RESQUE: A Resource Based Simulation System for Construction Process Planning." Ph.D. Dissertation, University of Michigan, Ann Arbor, MI.

Cor, H. (1998). "Using Simulation to Quantify the Impacts of Changes in Construction Work." Master's Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Damrianant, J., and Wakefield, R. R. (1997). "A Petri-Net Based System for Modeling and Computer Simulation of Construction Operations", *Proceedings of the 4th Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 190-197.

Evans, J. B. (1989). *Structures of Discrete Event Simulation*, John Wiley & Sons, New York, NY.

Gonzalez-Quevedo, A., AbouRizk, S. M., Iseley, D. T., Halpin, D. W. (1993). "Comparison of Two Simulation Methodologies in Construction", *Journal of Construction Engineering and Management*, ASCE, 119(3) 573-589.

Gonzalez-Quevedo, A. (1995). "Sensitivity Analysis of Construction Simulation", *Proceedings of the 2nd Congress on Computing in Civil Engineering*, ASCE, Reston, VA 1427-1434.

Halpin, D. W. (1977). "CYCLONE - Methods for Modeling Job Site Processes", *Journal of the Construction Division*, ASCE, 103(3). 489-499.

Halpin, D. W. (1990). *Micro-CYCLONE User's Manual*, Department of Civil Engineering, Purdue University, W. Lafayette, IN.

Halpin, D. W., and Riggs, S. (1992). *Design of Construction and Process Operations*, John Wiley & Sons, New York, NY.

Halpin, D. W., and Woodhead, R. (1976). *Design of Construction and Process Operations*, John Wiley & Sons, New York, NY.

Harmelink, D. J., and Bernal, M. A. (1998). "Simulating Haul Durations for Linear Scheduling" *Proceedings of the 1998 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 1607-1613.

Harris F. C., and Evans, J. B. (1977). "Road construction - Simulation Game for Site Managers", *Journal of Construction Engineering and Management*, ASCE, 103(CO3). 405-414.

Hills, P. R. (1971). "*HOCUS*", P. E. Group, Egham, Surrey, England.

Hooper, J. W. (1986). "Strategy Related Characteristics of Discrete-Event Languages and Models", *Simulation*, Simulation Councils Inc., 46:4 153-159.

Hooper, J. W. and Reilly, K. D. (1982). "An Algorithmic Analysis of Simulation Strategies", *International Journal of Computer and Information Sciences*, 11 2 101-122.

Huang, R. Y. and Halpin, D. W. (1994). "Visual Construction Operations Simulation - The DISCO Approach", *Journal of Microcomputers in Civil Engineering*, 9 (1994). 175-184.

Ioannou, P. G. (1989). *UM-CYCLONE Reference Manual*, Technical Report UMCE-89-11, Department of Civil Engineering, University of Michigan, Ann Arbor, MI.

Ioannou, P. G., and Martinez, J. C. (1996a). "Animation of Complex Construction Simulation Models," *Proceedings of the 3rd Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 620-626.

Ioannou, P. G., and Martinez, J. C. (1996b). "Comparison of Construction Alternatives Using Matched Simulation Experiments," *Journal of Construction Engineering and Management*, ASCE, Vol. 122, No. 3, 231-241.

Ioannou, P. G., and Martinez, J. C. (1996c). "Simulation of Complex Construction Processes," *Proceedings of the 1996 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 1321-1328.

Ioannou, P. G., and Martinez, J. C. (1996d). "Scaleable Simulation Models for Construction Operations," *Proceedings of the 1996 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 1329-1336.

Jensen, K. (1992). *Coloured Petri Nets*, Volume 1: Basic Concepts, Springer-Verlag, New York, NY.

Kalk, Anthony. (1980). "INSIGHT: Interactive Simulation of Construction Operations Using Graphical Techniques", Technical Report No. 238, Department of Civil Engineering, Stanford University.

Law, A. M., and Kelton, D. K. (1991). *Simulation Modeling and Analysis*, 2nd ed. McGraw-Hill, New York, NY.

Lin, J. T., and Lee, C. C. (1993). "A Three-Phase Discrete Event Simulation with EPNSim Graphs", Simulation, Simulation Councils Inc., 60:6, 382-392.

Liu, L. Y. (1991). "COOPS: Construction Object-Oriented Simulation System." PhD Dissertation, University of Michigan, Ann Arbor, MI.

LLuch, J., and Halpin D. W. (1982). "Construction Operation and Microcomputers", *Journal of the Construction Division*, ASCE, 108(CO1). 129-145.

Lutz, J. D., Halpin, D. W., and Wilson, J. R. (1994). "Simulation of Learning Development in Repetitive Construction", *Journal of Construction Engineering and Management*, ASCE, 120(4). 753-773.

McCahill, D. F., and Bernold, L. E., (1993). "Resource-Oriented Modeling and Simulation in Construction", *Journal of Construction Engineering and Management*, ASCE, 119(3). 590-606.

Martinez, J. C. (1996). "STROBOSCOPE: State and Resource Based Simulation of Construction Processes." PhD Dissertation, University of Michigan, Ann Arbor, MI.

Martinez, J. C. (1997). "Structure of an Advanced Course in Computer Applications and Simulation in Construction," *Proceedings of the 4th Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 206-215.

Martinez, J. C. (1998). "EarthMover - Simulation Tool for Earthwork Planning," *Proceedings of the 1998 CIB W78 Conference*, Department of Construction Management and Economics, Royal Institute of Technology, Stockholm, Sweden.

Martinez, J. C., and Ioannou, P. G. (1995). "Advantages of the Activity Scanning Approach in the Modeling of Complex Construction Processes" *Proceedings of the 1995 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 1024-1031

Martinez, J. C. and Ioannou, P. G. (1996). "State-Based Probabilistic Scheduling Using STROBOSCOPE's CPM Add-On." *Proceedings of Construction Congress V*, ASCE, Reston, VA, 438-445.

Martinez, J. C., Ioannou, P. G., and Carr, R. I., (1994). "State and Resource Based Construction Process Simulation", *Procedings of the First Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 177-184.

Moavenzadeh, F., and Markow, M. (1976). "Simulation Model for Tunnel Construction Costs", *Journal of the Construction Division*, ASCE, 102(CO1). 417-432.

Mohieldin, Y.A. (1989). "Analysis of construction processes with non-stationary work task durations." Ph.D. Dissertation, University of Maryland.

Oloufa, A. A., and Ikeda, M. (1997). "Library-Based Simulation Modeling in Construction", *Proceedings of the 4th Congress on Computing in Civil Engineering*, ASCE, Reston, VA, 198-205.

Odeh, A. M. (1992). "Construction Integrated Planning and Simulation Model." Ph.D Dissertation, University of Michigan, Ann Arbor, MI.

Petri, C. A. (1966). "Communication with Automata", Technical Report, Griffiss Air Force Base, RADC-TR-65-377, 1, 1.

Pritsker, A. A., and Happ, W. W. (1966). "GERTS: Part 1 – Fundamentals", *Journal of Industrial Engineering*, AIIE, 17(5), 267-274.

Pritsker, A. A. (1986). *Introduction to Simulation and Slam II*, 3rd ed. Halsted Press, John Wiley & Sons, New York, NY.

Sagert, Per (1995). "Simulation of Construction Operations." Master's Thesis, Chalmers University of Technology, Göteborg, Sweden.

Sangarayakul, B. (1998). "Use of Simulation to Analyze Block Manufacturing Methods." Master's Thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Sawhney, A. (1997). "Petri Net Based Simulation of Construction Schedules", *Proceedings of the 1997 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 1111-1118.

Schruben, L. W. (1995). "Building Reusable Simulators Using Hierarchical Event Graphs", *Proceedings of the 1995 Winter Simulation Conference*, Society for Computer Simulation, San Diego, CA, 472-475.

Shi, J., AbouRizk, S. (1997). "Resource-Based Modeling for Construction Simulation", *Journal of Construction Engineering and Management*, ASCE, 123(1). 26-33.

Stroustrup, B. (1998). *The C++ Programming Language*, 3rd Edition, Addison-Wesley, Reading, MA.

Szymankiewicz, J., McDonald, J. and Turner, K., (1988). *Solving Bussiness Problems by Simulation*, McGraw-Hill, London.

Tocher, K. D. (1963). *The Art of Simulation*, English University Press, London.

Tocher, K. D. (1979). "Keynote Address", *Proceedings of the 1979 Winter Simulation Conference*, IEEE, New York, NY, 641-654.

Tocher, K. D. (1964). "Some Techniques of Model Building", *Proceedings of IBM Scientific Computing Symposium on Simulation Models and Gaming*, IBM, New York, 119-155.

Tocher, K. D., and Owen, D. G. (1960). "The Automatic Programming of Simulations", *Proc. IFORS Conference*, IFORS, Aix-en-Provence, 50-67.

Tommelein, I. D. (1998). "Pull Driven Scheduling for Pipe-Spool Installation: Simulation of Lean Construction Technique", *Journal of Construction of Engineering and Management*, ASCE, 124(4). 279-288.

Touran, A., and Asai, T. (1987). "Simulation of Tunneling Operations", *Journal of Construction of Engineering and Management*, ASCE, 113(4). 554-568.

Wang, W. C. (1996). *"Model for Evaluating Networks Under Correlated Uncertainty - NETCOR,"* Ph.D. Dissertation, University of California, Berkeley, CA.

Woods D. G., and Harris F. C. (1980). "Truck Allocation Model for Concrete Distribution", *Journal of the Construction Division*, ASCE, 106(CO2). 131-139.

Zeigler, B. P. (1976). *Theory of Modeling and Simulation*, John Wiley & Sons, New York, NY