Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

COMPARISON OF SIMULATION MODELING TECHNIQUES THAT USE PREEMPTION TO CAPTURE DESIGN UNCERTAINTY

Nuno Gil
Iris D. Tommelein

University of California
Construction Engineering and Management Program
Department of Civil and Environmental Engineering
215 McLaughlin Hall
Berkeley, CA 94720-1712, U.S.A.

ABSTRACT

This paper describes a process, implemented using two simulation engines that adopt, respectively, the event scheduling paradigm and the activity scanning paradigm. The process being modeled is design development in an unpredictable environment. Unpredictability means that criteria are prone to change during design, thereby interrupting ongoing work and causing design iteration. Probability density curves, input to the simulation, capture uncertainties regarding design criteria during the development of R&D semiconductor fabrication facilities. The simulation of process changes calls for preempting tasks or events, and scheduling new tasks or events. The implementations in alternative modeling paradigms illustrates the use of a top-down vs. a bottom-up approach in process modeling. The two engines that were used, SIGMA and STROBOSCOPE, both are programmable so that the model could be implemented without difficulty in either one.

1 INTRODUCTION

Analytical models have yielded managerial insight into design development processes unfolding in unpredictable environments (e.g., Krishnan et al. 1997, Bhattacharya et al. 1998). Unpredictability has been modeled in diverse ways, such as by assuming design changes in preliminary information, or by assuming faults in the information resulting from upstream tasks. The goal of these models is to provide frameworks that help practitioners determine how to best exchange information, if changes in preliminary information are anticipated. By and large, these models claim that formulating a sharp product definition early on may not be desirable or even feasible for product development in unpredictable environments. Instead, they advocate that firms delay commitments and allow real-time definition along the development process, according to the level of uncertainty they expect, their own risk profile, and the value of customer information. These findings are confirmed by empirical studies on new product development in unpredictable environments (e.g., Iansiti 1995, Eisenhardt and Tabrizi 1995, Ward et al. 1995, Thomke and Reinertsen 1998).

Aside from the issue on how to characterize and model unpredictability in design, this paper describes the use of two computer simulation engines to model a design process. SIGMA uses event scheduling whereas STROBOSCOPE uses activity scanning. We compare and contrast our subjective assessment of the ease with which these tools enable the modeler to represent the chosen process and to capture process characteristics. This paper complements Gil et al.'s (2001) paper in this 2001 WSC conference, that uses SIGMA to study the effects of postponed commitment strategies to manage design development processes in unpredictable environments.

2 SELECTION OF TWO SIMULATION ENGINES

2.1 Event Scheduling vs. Activity Scanning

Event scheduling and activity scanning are two major modeling paradigms used by discrete-event simulation packages. A third paradigm is based on the use of block languages, but this alternative is not discussed here.

Event scheduling systems focus on the concept of an event graph, comprising vertices and edges. Vertices are associated with state changes. Edges are associated with conditions and delays. Event scheduling systems model a system as it evolves over time by "identifying its characteristic events and then writing a set of event routines that give a detailed description of the state changes taking place at the time of each event" (Law and Kelton 2000 p. 205). SIGMA (Schruben and Schruben 1999), the event scheduling engine used in this comparison, provides fundamental, low-level

programming language constructs on which higher-level constructs can be built. SIGMA can be used to model problems in any domain. It has been used in diverse applications, including queuing problems, scheduling problems, as well as systems dynamics problems such as the growth and decline of biological populations (e.g., Duenyas et al. 1994, Allore et al. 1998).

Activity scanning systems provide mathematical and graphical modeling techniques that focus on the operating cycles of resident entities (physical or abstract resources). STROBOSCOPE (Martinez 1996), the activity scanning engine used in this comparison, provides several higher-level programming language constructs. STROBOSCOPE can be also used to model problems in any domain. The program was created more recently and has since been used mainly to model queuing problems in architecture-engineering-construction (AEC) at large, including design and construction operations (e.g., Ioannou and Martinez 1996) and exchanges of information to support construction management (e.g., Tommelein 1998).

Preemption is an action taken to check another action beforehand. Preemption is required to model either the cancellation of a scheduled activity before it has started because of an event that occurs beforehand, or the interruption of an activity because of an event that occurs during its execution. Preemption is useful, for instance, to simulate disruptions caused by machine breakdowns, to reschedule tasks because of (un)anticipated events (e.g., the expected mean time to failure is less than the planned activity completion time), or to release or draw resources into an activity during the activity's execution (e.g., when it is discovered that some resources are lacking).

Event scheduling systems such as SIGMA have the ability to model both the process flows of transient entities as well as the operating cycles of resident entities. SIGMA's graphical interface also includes scheduling edges and canceling edges, which makes it easy for users to build a model that can interrupt and cancel tasks in the course of a simulation run. Thus, preemption is easily modeled. Ingalls et al. (1996) present alternative ways to model preemption using event graphs in SIGMA without using canceling arcs. Nevertheless, they acknowledge the convenience and functionality of the 'canceling edge' construct.

STROBOSCOPE does not contain explicit language constructs to model preemption. This was the result of careful deliberation by its developers. Preemption occurs in many systems beyond the most simple ones. Expressing the specifics of a case of preemption in a simulation language requires much more than merely interrupting an activity. It may require selecting one or a few instance(s) to interrupt among multiple instances of the same activity, or drawing one or a few resource(s) out of selected instances of multiple activities. Preemption may manifest itself differently for different instances of activities and resources. Numerous possibilities also exist regarding how to proceed with the simulation after preemption has occurred. Capturing useful cases of preemption in higher-language constructs is feasible. Nevertheless, given the inevitable complexity of those constructs if they were to capture any preemption subtlety at all, it is not obvious that learning to use them and then using them would make it any easier on the programmer than implementing the desired preemption mechanism using the already existing constructs. Consequently, users of today's STROBOSCOPE version 1.5.3.0 must exert special effort in terms of code writing to implement preemptive behavior.

3 DESCRIPTION OF DESIGN PROCESS TO BE MODELED

3.1 Example Process that Requires Preemption

The model presented in this paper greatly simplifies the complexity of design development processes unfolding in unpredictable environments. Unpredictability here means that design criteria changes—hard to anticipate—may occur during design. Gil (2001) studied this process in greater depth, specifically for design development of a semiconductor fabrication facility (fab).

Figure 1 presents a schematic process model for design development. Design development comprises two distinct phases: initial conceptualization followed by concept development. During conceptualization, designers primarily use empirical rules and historical data to take a first pass at the design parameters. During concept development, they use sophisticated analytical tools to refine the decisions made during conceptualization. Concept development starts immediately after the end of conceptualization. Gil et al. (2001) also studied process impacts that result from postponing the start of concept development.

Figure 1: Design Development Model

Internal and external conditions may force designers to repeat these two phases. Designers may reiterate the process in their search for a satisfying solution if time allows (Simon 1969). Iteration may happen for the sake of learning through exploration, even if designers possessed all the information they needed from the start. For simplicity's sake, we based our model on the assumption that designers would perform work only once to find a satisfying solution provided that design criteria did not change. That is, there is no repeated search for alternatives. Design iterations may also be caused by interdependencies with work being executed by other design specialists. We do not capture design iteration due to concurrency of design processes across disciplines in this

model. Instead, we focus on the impact external (client-driven) changes have on the design process and we assume that those changes arise irrespectively of the state of the on-going design process.

3.2 Uncertainty

Uncertainty in fab design criteria stems from factors such as the concurrency of the fab design effort with the chip product development, the unknown characteristics of the production tools, the possibility of a change in fab layout, and the unpredictability of market demand. In this paper, we describe only design criteria changes caused by uncertainty, namely the changes in the dimensions of the cleanroom (the cleanroom is the space inside a fab where the chips are produced). Discussing only one type of change suffices to show alternative ways for modeling preemption in SIGMA and STROBOSCOPE. Gil et al. (2001) study a more complex process that also includes changes in tools to be housed in the fab.

Cleanroom dimension changes, although not frequent, occur if the manufacturer unexpectedly needs to increase (or decrease) the fab capacity. We assume that when the cleanroom width and length change, AEC designers have to rework the conceptualization and the concept development phases. The probabilistic and temporal relationships between consecutive cleanroom changes were developed jointly with practitioners and are mathematically stated as

$$P(\text{change}_1) = A \tag{1}$$

$$P(\text{change}_2 \mid \text{change}_1) = \frac{A}{1+B} \tag{2}$$

$$P(\text{change}_3 \mid \text{change}_2) = \frac{A}{1+2.0*B} \quad ,\cdots, \tag{3}$$

$$P(\text{change}_i \mid \text{change}_{i-1}) = \frac{A}{1+(i-1)*B}, \quad i \geq 2 \tag{4}$$

$$P(\text{change}_i \mid \overline{\text{change}_{i-1}}) = 0, \quad i \geq 2 \tag{5}$$

$$T_i = C*\left[i + \sum_{s=1}^{i}(1+(s-1)*B)*\text{beta}_i(2,2)\right], \; i \geq 1 \tag{6}$$

where

P(i)	:	Probability of change i to occur
P(i|i-1)	:	Probability of change i to occur, given the prior occurrence of change i-1
A, B, C	:	Constants
T_i	:	Time when change i occurs [days]
$\text{beta}_i(\alpha_1=2,\alpha_2=2)$	:	Symmetric beta random variable that is sampled for every value of i

We used rescaled and relocated symmetric beta random variables [a+(b-a)*beta(α_1=2,α_2=2)] to express the variability around the time when a change can occur. The probability of occurrence of a first change is higher than the probability of occurrence of a subsequent change, and earlier changes are likely to occur within a more narrow range of values than later ones. Accordingly, the model decreases the probabilities of the subsequent changes by dividing the probabilities of the first change by the terms of an increasing numeric sequence. In addition, the model increases the rescaled intervals of the beta distributions (b-a) between subsequent changes by multiplying them by those same numbers. To clarify, the probability of occurrence of a stream of changes is

$$P(\text{change}_i \cap \text{change}_{i-1} \cap \ldots \cap \text{change}_1) = \prod_{s=1}^{i} \frac{A}{1+(s-1)*B}, \; i \geq 1 \tag{7}$$

Practitioners estimated, for the specific case of cleanroom changes, A and B equal to 0.5, and C equal to 20 days. These estimates reflect their perceptions of the frequency and time of occurrence of cleanroom dimensions changes, for the case of research and development fabs of complex process technologies. Figure 2 depicts an excerpt of the probabilistic tree that is the basis for the probability density curves for changes in cleanroom dimensions. Figure 3 illustrates these probability density curves with data points derived from 1,000 simulation runs.

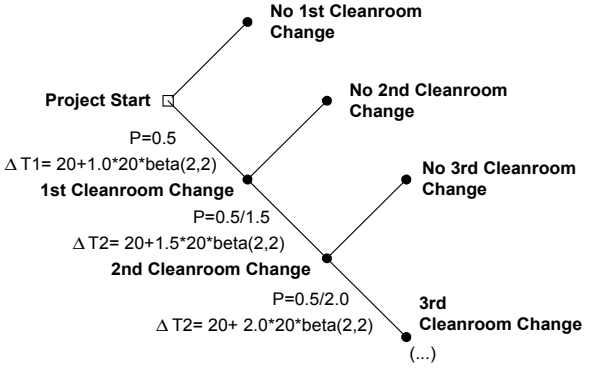

Figure 2: Excerpt of Detailed Probabilistic Tree for Cleanroom Dimensions Change

3.3 Rework

Hopp and Spearman (1996 p. 362) discuss rework in the context of factory physics. They assume that a machine produces defective parts and they then study the effects of reworking those parts in terms of production line performance. With the help of computer simulation, they demonstrate the negative consequences of rework to cycle time, throughput, and work in process. They conclude that the longer the rework loop, the more pronounced these consequences are.

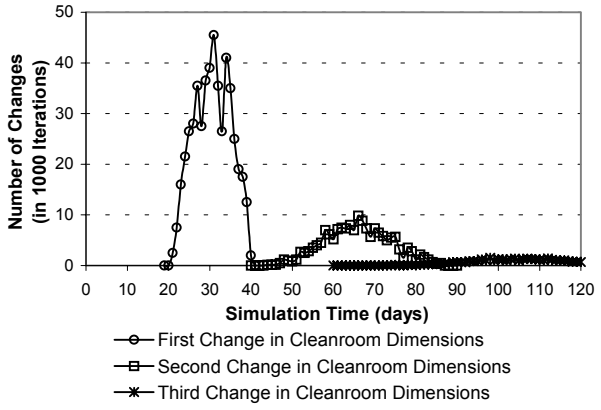

Figure 3: Histograms of Changes in Cleanroom Dimensions for 1,000 Simulation Runs

In this work, we used a simple rework algorithm that posits that whenever a change occurs, the expected duration for a design phase that needs to be reiterated is equal to its initial duration. In other words, the algorithm assumes that designers do not learn or gain any process efficiency from one iteration to the next. This scenario is assumed to hold both for work interrupted when a change occurred and for work that was already done when a change occurred. It can be written as

$$D_{i+1} = D_i = D_1, \forall i \quad (8)$$

where

- i : number of times designers start to perform the task ($i = 1, 2, 3, \ldots$)
- D_1 : expected duration of a task in the first iteration, if no interruptions occur [days]
- D_i : expected duration of a task in the last iteration, irrespectively of whether or not the task was interrupted in a previous iteration [days].

3.4 Additional Assumptions

To provide clarity and to make it easier to interpret the model's result, we made the following assumptions:

- Each task has a deterministic duration. Computer simulation lends itself to express stochastic durations, but given the sequential nature of this specific process, stochastic behavior would not influence the average results of the performance variables that were obtained with the deterministic model (this is a consequence of the Central Limit Theorem).
- Design criteria changes scheduled to occur after the end of concept development are not considered.

4 MODEL IN SIGMA

We first implemented the process in SIGMA. Figure 4 illustrates the corresponding event graph model. The geometric figures represent events. Rectangles with a cut-off corner represent the beginning or end events of design phases. Circles represent the START and END of the design development process. The diamond represents a CLEANROOM CHANGE [dimensions]. Arrows represent relationships between the events they connect. Associated with each arrow is a set of conditions. A solid arrow denotes that the event from which it emanates causes the event to which it points to occur after a time delay greater than or equal to zero. Similarly, a dashed arrow causes the destination event to be cancelled after some time delay.

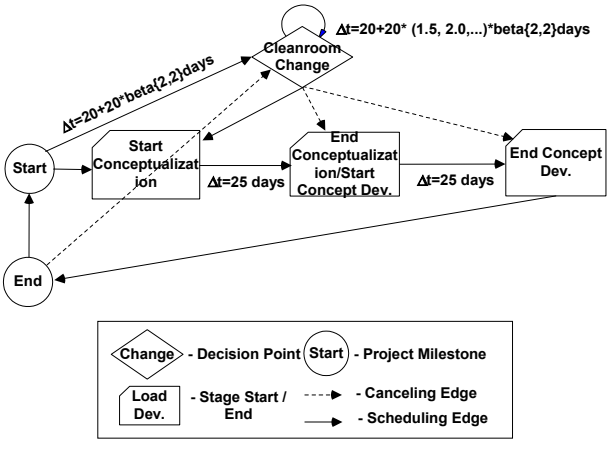

Figure 4: Event Graph Model for Design Process Development

The design process simulation START[s] (labels of symbols in figures are capitalized in the text) by scheduling the START [of the] CONCEPTUALIZATION phase. The START event also schedules, with probability A, that a first CLEANROOM CHANGE will occur after a stochastic time delay. When a CHANGE event occurs, it may schedule a subsequent CHANGE. The END [of the] CONCEPTUALIZATION phase coincides with the START [of the] CONCEPT DEVELOPMENT phase. The END [of] CONCEPT DEVELOPMENT schedules an END event for the simulation run. This END event then collects the values of the performance variables (e.g., total project duration; number of days spent on conceptualization or on concept development, including rework; number of iterations of each task) and it cancels any changes that are still scheduled to occur. This END event can be programmed to schedule a START event for a new, stochastically independent simulation run should multiple iterations of the model be needed to generate statistics about those performance variables.

At the heart of the simulation model in SIGMA is the use of canceling relationships between events. When the simulation executes a canceling relationship, it cancels the destination events that were previously scheduled. Accordingly, a CLEANROOM CHANGE will immediately cancel all the scheduled design events and schedule a new START [of] CONCEPTUALIZATION. CONCEPTUALIZATION lasts 25 days. Should any change interrupt it, designers would have to repeat that effort. Designers start CONCEPT DEVELOPMENT on day 25 if no cleanroom changes had yet occurred, or later, after CONCEPTUALIZATION ends if changes occurred in the mean time.

5 MODEL IN STROBOSCOPE

We then implemented the process in STROBOSCOPE. Since this could require some programming through the graphical interface, we thoughtfully developed two alternatives. The first alternative exploits a characteristic of the process as it had been formulated. Specifically, since all change events are assumed to occur independently of the state of the design (only external changes are being considered), the simulator can generate that stream of changes first, then use the resulting stream of event times as input to the simulation of the design process. This implementation is further detailed in sections 5.1 Change Events and 5.2 Design Process. The second alternative implements a more general form of preemption, including cases where changes—causes for preemption—become known during the execution of design activities. This is detailed in section 5.3 More General Model for Preemption in STROBOSCOPE.

5.1 Change Events

The occurrence of change events as depicted in the event tree (Figure 2) is captured by the STROBOSCOPE model shown in Figure 5. Geometric figures represent holding places for resources or time delays. Rectangles represent activities that take some duration to be executed. Rectangles with a cut-off corner are a special kind of activity (combination activity or combi) in that they require resources drawn from one or several queues as input. Circles with a tail represent queues where resources wait until being drawn into a combi. Arrows express precedence logic and resource flow. The circle with a triangle is a fork, from which resources will flow along one of multiple arrows emanating from it, as decided by probabilistic sampling or by a decision rule.

START is a dummy activity with zero duration. Branching of the tree is achieved using a probabilistic fork OTHER CHANGE. If p is the probability for not having a subsequent change and thus following arrow c6, then $1\text{-}p$ must be the probability for having a subsequent change and thus following arrow c3. This probability p varies with

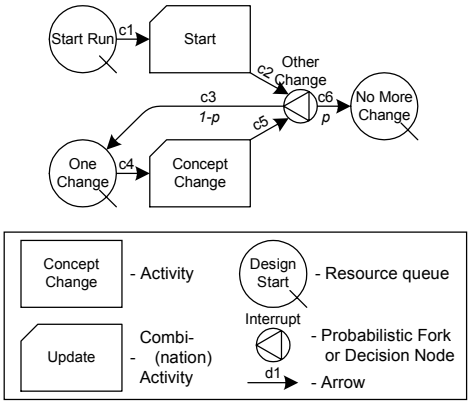

Figure 5: STROBOSCOPE Process Model to Generate Change Events

each iteration and a new value is computed after each CONCEPT CHANGE has taken place. During each instance of CONCEPT CHANGE the simulation engine samples the time that will elapse until the next change event occurs. This model uses an array to store the event times of changes. This array is then input to the model of the design process.

5.2 Design Process

The one-way dependence between the change-event stream and the design process makes preemption of design tasks simple to mimic. When STROBOSCOPE starts an activity and samples its duration from the user-specified probability distribution (in this simple process, the duration of CONCEPTUALIZATION and of CONCEPT DEVELOPMENT is assumed to be constant at 25 days), it can compare that duration against the time until the next change occurs. The smaller of the two determines the simulated task's duration.

CPTDUMMY and DVTDUMMY perform this sampling but they are dummy combis with zero duration. The comparison is done in the folk that follows each of them (Figure 6) and according to the outcome, the process is routed along one or the other arrow (mutually exclusively d3 XOR d5) emanating from it.

Taking advantage of the process characteristics, the implemented model is very simple and also efficient in runtime.

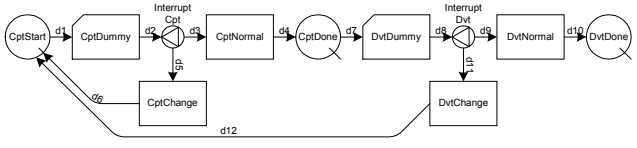

Figure 6: STROBOSCOPE Process Model for Design, including Conceptualization and Concept Development

5.3 More General Method for Preemption in STROBOSCOPE

A more general method for modeling preemption is shown in Figure 7. This method does not exploit the specific characteristics of the process at hand, which allowed for decoupling (Figures 5 and 6) and consecutive execution of the change process model and the design process model.

The underlying idea for the more general method is simple. Preemption means canceling an event that was scheduled to occur at some time in the future, and that, accordingly, is posted on the simulation engine's future event list (FEL). In STROBOSCOPE, as in many other simulation engines, the FEL is internal to the program and the user cannot manipulate its data directly. To fake access, the user can mimic the FEL by creating their own. A FEL is essentially a queue of events, so the STROBOSCOPE queue construct suits this purpose. In contrast, SIGMA users can access the FEL directly (though this feature is turned off in the 'student' version of the program), so that no faking would be necessary.

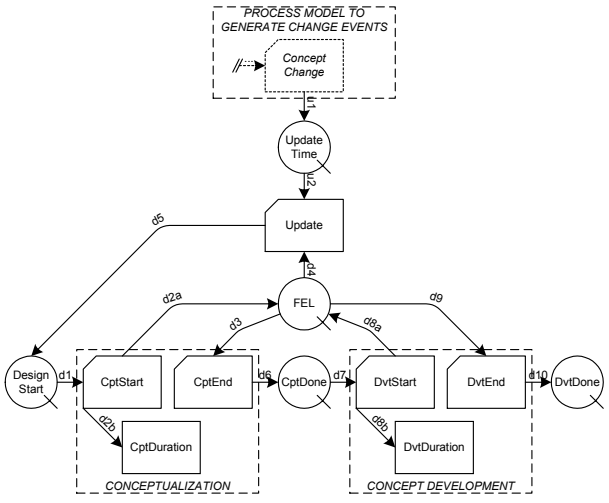

Figure 7: More General STROBOSCOPE Process to Model Design with Preemption

Tasks are broken down in three parts: (1) their start, (2) their execution with the normal duration, and (3) their end. (1) and (3) are modeled as combis with a zero duration, so they are like events. CPTSTART (DVTSTART) releases an entity to the shown FEL queue, which is like posting an event on the future event list. This entity has an attribute 'endtime' with as value the simulation time at which it is created plus the normal duration of (2). That is, the value of 'endtime' is the design task's end time should that task complete without being interrupted by changes.

CPTSTART (DVTSTART) initiates CPTDURATION (DVTDURATION). The end of the latter activity will post an event at the time of normal completion on the simulator's internal FEL. In turn, CPTEND (DVTEND) is programmed to start when an entity is available on the shown FEL queue whose attribute 'endtime' matches the end of CPTDURATION (DVTDURATION). When advancing the internal simulation clock, the simulator will evaluate this condition and find it to be true when CPTDURATION (DVTDURATION) completes normally, that is, no changes occurred in the mean time.

The stream of changes is modeled in the same way as was explained previously (the dashed box at the top of Figure 7 represents the process shown in Figure 5). However, the change process here is an integral part of the process model. Changes are generated as the simulation of the design process progresses. Every time a change arises, the UPDATE activity will draw all entities from the FEL (arrow d4) and restart the simulation (arrow d5).

This model clearly is more complex because the STROBOSCOPE user must program their own FEL and the desired interruption of activities and routing of resources. Modeling the FEL (or at least some of its features) penalizes the simulation at runtime.

6 DISCUSSION

Modeling necessarily is subjective. There is no unique 'right' way to model a system. Different programmers using SIGMA or STROBOSCOPE are likely to create different implementations for the preemption of design processes as described in this paper. The observations that follow reflect the authors' conceptualization of the process, their prior knowledge of the simulation engines, individual experience and skill in taking advantage of various simulation capabilities, and personal preference.

6.1 Ease of Implementation

Event scheduling nicely matches some people's mental models of processes evolving over time as a dynamic succession of events. Other people may think more intuitively in terms of processes comprising activities that require resources and then take time to execute. The authors do not argue in favor of using one or the other modeling paradigm.

SIGMA implements the event scheduling paradigm by characterizing edges as being of one of three types: (1) canceling, (2) pending, and (3) scheduling, which Schruben and Schruben (1999) provide more detailed information on. This expressiveness of SIGMA to cancel and reschedule tasks makes the implementation of preemption straightforward, whether or not mutual interdependence exists between design tasks and the stochastic stream of changes.

STROBOSCOPE does not have explicit constructs to model preemption but the user can model systems so as to mimic event scheduling as needed (e.g., Figure 7).

6.2 Graphical Interface

SIGMA provides a graphical interface for the user to develop the model (Figure 4). Underlying the nodes (vertices) and arcs (edges) shown graphically, are interface windows where the programmer has to enter code.

The SIGMA programmer has to explicitly spell out all state changes when an edge gets cancelled. In addition, at each state change, the programmer has to collect performance data depending on the variables being tracked. Using SIGMA's educational version, this coding may be rather cryptic.

STROBOSCOPE comes with graphical templates to be uploaded in Visio (Microsoft 2001). As is the case with SIGMA, this interface enables the user to develop the model (Figures 5, 6, and 7). STROBOSCOPE's interface windows provide more fields than SIGMA's, so this may expedite model development. Nevertheless, the sophisticated STROBOSCOPE user may also (have to) program rather complex constructs in order to obtain the desired behavior of a model.

Either program's interface simply constitutes of fields for the user to enter data and code. Closing the interface box formats that input so that it can be read by the simulation engine. These graphical interfaces are often used for preliminary prototyping of a system and to generate code. Rather than continuing to use the graphical interface throughout all experimentation with the model, program users are likely to resort to editing that code directly. This enables them to see, in lexical order, the input data, variable definitions, formulas, and so on, that are not shown all simultaneously in the graphical interface.

6.3 Top-Down vs. Bottom-Up Language Constructs

SIGMA and STROBOSCOPE are both general purpose simulation engines. They can be applied to model a diverse range of processes in various domains. They differ however in the degree of programming complexity of the constructs they provide from the start. SIGMA provides lower-level generic constructs—essentially the vertex and the edge—that require little time to learn to use. These constructs make SIGMA flexible for modeling complex processes, provided that the user masters the underlying logic of programming the state changes at the vertices and the Boolean conditions at the edges.

A powerful feature is that SIGMA users can group and save SIGMA models as independent user tools. Each event graph has clear inputs and outputs so that linking them at a higher level does not create ambiguity. The resulting tools can be made available on SIGMA's graphical template list for later integration in more complex models. In doing so, users leverage SIGMA's lower-level constructs as customized higher-level constructs, making the program's functionality resemble that provided by higher-level programming languages.

STROBOSCOPE provides higher-level programming language constructs from the start. STROBOSCOPE therefore takes more time to learn but doing so is worthwhile when those constructs suit the domain of application and the user gets to apply their knowledge when modeling processes that intuitively match those constructs. However, higher-level constructs are tedious to use when lower-level functionality is to be achieved, as was the case in modeling the chosen process with preemption.

7 CONCLUSIONS

This paper has presented the implementation of a design process with tasks that were preempted. This process was implemented several times by the authors, once using the SIGMA simulation engine that follows the event-scheduling paradigm, and twice using the STROBOSCOPE simulation engine that follows the activity-scanning paradigm. This effort illustrates that both paradigms can support the modeling needs of this specific process. It confirms that alternative ways exist to model the process within each paradigm. The authors have found it very useful, from a model validation perspective, to re-implement the model that was first developed in SIGMA using another simulation language, in this case STROBOSCOPE. For instance, this effort highlighted several modeling assumptions that otherwise would have been taken for granted.

Not all event scheduling or activity scanning simulation languages provide explicit constructs to describe the preemption of tasks. In real-life systems, including physical as well as social systems, tasks often get interrupted or cancelled altogether. Simulations that model preemption therefore get one step closer to the actual behavior of real-world systems.

The requirements for preemption as modeled here are rather simple. The stream of changes is stochastically independent of the execution of the design process. In addition, preempted activities simply cause the design process to restart and there is no need to reroute resources such as people or equipment, as the model does not include any such resources. In practice, preemption mechanisms may be more complex and require significant managerial decision making.

ACKNOWLEDGMENTS

This research was funded by grant SBR-9811052 from the National Science Foundation, whose support is gratefully acknowledged. Any opinions, findings, conclusions, or recommendations expressed in this report are those of the authors and do not necessarily reflect the views of the National Science Foundation. Financial support from the Por-

tuguese Foundation of Science and Technology, through a scholarship awarded to Nuno Gil, is also gratefully acknowledged.

Last, but not least, we thank Lee Schruben, as well as Photios Ioannou and Julio Martinez for sharing their enthusiasm for SIGMA and STROBOSCOPE, respectively, and explaining programming features we would otherwise not have appreciated.

REFERENCES

Allore, H. G., Erb, H. N., Schruben, W. L., and Oltenacu, P. A. (1998). "Design and Validation of a Dynamic Discrete Event Stochastic Simulation Model of Mastitis Control in Dairy Herds." *Journal of Dairy Science*, 81(3), 694-702.

Bhattacharya, S., Krishnan, V., and Mahajan, V. (1998). "Managing New Product Definition in High Velocity Environments." *Management Science*, 44 (11).

Duenyas, I., Fowler, J., and Schruben, L. (1994). "Planning and Scheduling in Japanese Semiconductor Manufacturing." *Journal of Manufacturing Systems,* 13, 323-332

Eisenhardt, M. K. and Tabrizi, B. N. (1995). "Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry." *Admin. Sci. Qtrly.*, 40, 84-110.

Gil, N. (2001). *Product-Process Simulation to Support Contractor Involvement in Early Design*. Ph.D. Diss., Civil & Envir. Engrg. Dept., Univ. of Calif., Berkeley.

Gil, N., Tommelein, I. D., and Kirkendall, R. (2001). "Modeling Design Development Processes in Unpredictable Environments". *Proc. 2001 Winter Simulation Conference*. Invited Paper in the Session *"Extreme Simulation: Modeling Highly-Complex and Large-Scale Systems."*

Hopp, W. J. and Spearman, M. L. (1996). *Factory Physics. Foundations of Manufacturing Management.* The McGraw-Hill Companies, Inc., 668 pp.

Iansiti, M. (1995). "Shooting the Rapids: Managing Product Development in Turbulent Environments." *California Management Review*, 38 (1), Fall, 37-58.

Ioannou, P. and Martinez, J. (1996). "Comparison of Construction Alternatives Using Matched Simulation Experiments." *J. Constr. Eng. & Mgmt.,* ASCE, 122 (3), 231-241.

Ingalls, R. G., Morrice, D. J., and Whinston, A. B. (1996). "Eliminating Canceling Edges from the Simulation Graph Model Methodology." *Proc. 1996 Winter Simulation Conference*, 825-832.

Krishnan, V., Eppinger, S. D., and Whitney, D. E. (1997). "A Model-Based Framework to Overlap Product Development Activities." *Mgmt. Sci.*, 43 (4), 437-451.

Martinez, J. C. (1996). *STROBOSCOPE State and Resource Based Simulation of Construction Processes*. Ph. D. Diss., Civil & Envir. Engrg., Univ. of Michigan, Ann Arbor, MI, 518 pp. The STROBOSCOPE software is available for download from <http://www.strobos.ce.vt.edu/>.

Law, A. M. and Kelton, W. D. (2000). *Simulation Modeling and Analysis.* McGraw-Hill, Inc., 760 pp.

Microsoft (2001). Visio 2000. <http://www.microsoft.com/office/visio/> visited 7/1/01.

Schruben, D. A. and Schruben, L. W. (1999). *Event Graph Modeling using SIGMA*, Custom Simulations, <http://www.customsimulations.com> visited 7/1/01.

Simon, H. A. (1969). *The Sciences of the Artificial.* MIT Press (3rd edition 1996).

Thomke, S. and Reinertsen, D. (1998). "Agile Product Development: Managing Development Flexibility in Uncertain Environments." *Calif. Mgmt. Rev.*, 41 (1), 8-30.

Tommelein, I. D. (1998). "Pull-Driven Scheduling for Pipe-Spool Installation: Simulation of a Lean Construction Technique." *J. Const. Eng. & Mgmt.,* 124 (4), 279-288.

Ward, A., Liker, J. K., Cristiano, J. J., and Sobek II, D. K. (1995). "The Second Toyota Paradox: How Delaying Decisions can Make Better Cars Faster." *Sloan Mgmt. Review*, Spring, 43-61.

AUTHOR BIOGRAPHIES

NUNO GIL is a Doctoral Candidate of Construction Engineering and Management in the Civil and Environmental Engineering Department at the University of California, Berkeley. His research interests are in project-based operations management, systems engineering, lean management, and concurrent engineering. His work involves planning, scheduling, and simulation. His email and web addresses are <ngil@uclink4.berkeley.edu> and <http:// www.ce.berkeley.edu/~nunogil>.

IRIS D. TOMMELEIN is Professor of Construction Engineering and Management in the Civil and Environmental Engineering Department at the University of California, Berkeley. Her research interest is in developing principles of production management for projects in the architecture-engineering-construction industry, what is termed 'lean construction.' Iris has strong analytical, computational, and writing skills. She has a proven research track record that includes developing and documenting industry case studies for educational purposes. Her work involves computer-aided design, planning, scheduling, simulation, and visualization of construction processes. Her email and web addresses are <tommelein@ce.berkeley.edu> and <http://www.ce.berkeley.edu/~tommelein>.