

# Stochastic Simulation of Information Systems Designs from Data Flow Diagrams

James R. Warren, Jack W. Stott, and A. F. Norcio

*Department of Information Systems, University of Maryland, Baltimore, Maryland*

Many fundamental decisions in information system (IS) design are influenced by how quickly work must be done—where to provide computer support and the choice of hardware platform and data base management system—yet modeling the overall dynamics of designs is not a routine part of IS development. We believe that design performance modeling can be made more accessible to IS developers by allowing the specification of performance models via conventional data flow diagrams (DFDs) annotated with information on the performance of the DFD components. A prototype simulation environment has been developed that produces stochastic simulation results automatically from annotated DFDs. The prototype reads DFDs from a custom tool and formulates a corresponding simulation model. The prototype provides model-based expert advice in the use of the simulation model and in interpretation of its output. The use of the prototype is illustrated through its application to the quality assurance subsystem of a manufacturing operation.

## 1. INTRODUCTION

This paper describes a method for information system IS developers to assess the dynamic (i.e., performance-related) characteristics of IS designs. IS developers frequently benchmark the performance of the hardware and software components of proposed IS designs, but modeling the dynamics of the complete IS design is not a normal part of the system development process. This is unfortunate because IS dynamics are an important consideration in many IS design decisions:

1. The choice of hardware platform is determined largely on the basis of dynamics. Organizations

invest in mainframe computers because they believe they need the capacity and speed. A microcomputer attached to rewritable optical technology may offer the same online storage capacity as a mainframe system at far lower cost (one can even purchase products to attach numerous terminals to the microcomputer), but the microcomputer-based system lacks the speed of the main frame system for most real-world applications.

2. The choice of database management system (DBMS) architecture can be driven by dynamics. When first introduced, there was a great deal of concern that relational DBMS such as DB2 were too slow for production systems [1]. The first release of DB2 performed 1.65 to 2.35 times slower than the hierarchical IMS DBMS on IMS-type workloads [2]. A DB2-based IS is qualitatively different from an IMS-based system, possessing greater flexibility in modification of the DBMS schema and offering ad hoc query capability [3]. However, these advantages must be foregone if the IS requires the performance of the hierarchical DBMS. These qualitative trade-offs are increasingly common with the proliferation of DBMS products and fourth-generation languages that offer new advantages and capabilities for developers and end users.
3. IS dynamics influence the focus of automation: what to automate and to what extent. If a service-oriented company is overly slow in attending to customers, the organization's objectives are not best met by doubling the speed of the employee payroll system. The example application in section 5 illustrates an operation where a single data entry process is critical to the dynamics of the system.

A prototype system has been developed that automatically produces stochastic discrete event-based simu-

---

*Address correspondence to Professor A. F. Norcio, Dept. of Information Systems, University of Maryland, Baltimore, MD 21228.*

lation models from data flow diagrams (DFDs) and additional information regarding the performance of system components. That is, the DFD, a traditional notation used to express the direction and contents of normal information flows in proposed and existing systems, is given a new, nontraditional role via annotation with the dynamics of flow among system components. The goal is to provide IS design performance evaluation as an integral part of system development by allowing the production of simulation results directly from CASE tool data dictionaries. The prototype demonstrates the feasibility of this goal, because the drawing and annotation of DFDs are already standard CASE tool features.

We describe currently available fourth-generation languages (4GLs) and tools for the production of IS design dynamics simulation, discuss some of the difficulties in using stochastic simulation to make decisions, and describe the design of the prototype IS design dynamics simulation environment. The features of the prototype are illustrated through application to the quality assurance subsystem of an electronic component manufacturing operation. We conclude with remarks on the advantages the method offers to IS development in an integrated CASE environment.

## 2. PRODUCTION OF IS DESIGN DYNAMICS SIMULATIONS

Stochastic discrete event-based simulation of IS designs can be implemented using simulation 4GLs such as GPSS/H and SIMSCRIPT II.5, but most IS developers lack the time and expertise to write a simulation program every time a question arises about the dynamics of an IS design. The simulation modeling process can be supported by a simulation environment. TESS by Pritsker and associates is a simulation environment that supports the development of models in the SLAM II 4GL, allows specification of experimental conditions, and automatically stores results in a data base for report generation, graphical presentation, and animation [4].

Simulation 4GLs, even when encapsulated in simulation environments, are not designed specifically to support IS dynamic modeling. QASE RT by Advanced Systems Technologies supports modeling of wide- and local-area networks, where the application code size, type of hardware, and operating system of each computer in the network is considered. QASE RT allows the graphical depiction of software running on hardware so developers can study the performance of proposed systems. QASE RT is an improvement over general-purpose simulation 4GLs for the IS developer because it contains a great deal of knowledge about the dynamics of IS components and can formulate an inte-

grated dynamic model of any system specified by the user.

IS developers could make more use of dynamic modeling if no diversion from the IS development task were required. All the above tools require the developer to examine the proposed IS design in question and recode the elements of that design relevant to a dynamic analysis in an environment separate from the developer's CASE workbench. One effort to integrate dynamic analysis with the IS development environment predates CASE [5]. A variant on the problem statement language [6] was used to produce SIMSCRIPT programs directly from the system description. A prototype system has been developed to support the performance analysis of high-performance software as an integral part of the software development process [7]. This tool allows developers to consider the performance impact of implementational considerations (such as whether to use arrays or linked lists) in a machine-independent manner before writing code. Cadre Technologies is now introducing CASE tools such as Teamwork/SIM that have dynamic analysis capabilities via system simulation. These capabilities are distinct from diagram completeness and static requirements analysis functions offered by numerous CASE vendors [8].

## 3. USING STOCHASTIC SIMULATION TO MAKE DECISIONS

Stochastic (i.e., pseudorandom number-based) simulation has been widely used for system performance evaluation, but is often presented without attention to the statistical nature of simulation output [9]. The results of a single stochastic simulation run represent merely an observation from a population of possible simulation results. Stochastic simulation can never yield the outcome that a given parameter of the simulated system is exactly  $x$ . One must be content with a confidence interval; that is, "We are  $p\%$  confident that the true value lies between  $x - \delta$  and  $x + \delta$ ." One of the simplest methods of generating confidence intervals is the method of independent replications, where a simulation run is repeated a number of times to obtain a set of independent observations. For an unknown parameter  $\mu_x$  of the simulated system,  $P(\mu_x = \bar{X}(n) \pm t_{n-1, 1-\alpha/2} s) = 1 - \alpha$ , where  $\bar{X}(n)$  and  $s$  are the mean and standard deviation of the independent observations of  $\mu_x$ , respectively, and  $n$  is the number of replications. The important simulation run parameters in the replications method are the simulation duration, warm-up period (the amount of time the simulation is run and observations are discarded on the suspicion that the system has not yet reached a steady state), and the number of replications.

Expert systems have been developed to aid simulationists in the statistical issues of simulation [10]. Expert system support can include aid in determining simulation duration and number of replications, assessing the impact of initial conditions, constructing confidence intervals, comparing output from alternative designs, and estimating response/input variable relationships. Knowledge-based support can also be provided through online help embedded in a simulation environment. The help system for the simulation model development environment (SMDE) [11] includes: (1) a general assistance manager that offers an introduction to the SMDE, tutorials on usage of the tools in the environment, and a glossary of terms; and (2) help routines local to specific tools [12].

#### 4. A PROTOTYPE IS DESIGN DYNAMICS SIMULATION ENVIRONMENT

A prototype system has been developed to support the IS developer in the analysis of IS design dynamics with stochastic simulation. The prototype runs under the X Windows System on a Hewlett-Packard HP 9000 workstation. The environment presents the developer with three primary windows and various pop-up windows. In the upper left of Figure 1, the developer has a DFD-drawing utility. This allows the specification of system structure and dynamic attributes. A pop-up window allowing the annotation of dynamic attributes to a DFD process is shown in the lower left. In a true integrated CASE environment, DFD drawing should be performed using a middle-level CASE tool, where the DFD components are integrated with the IS design's data dictionary.<sup>1</sup> In the lower center portion of the screen is a simulation run parameter input window. This window allows the user to input the parameters to specify a method-of-replications simulation experiment. The prototype automatically interprets the DFD as a network of queues and conducts a simulation under the specified parameters. The user can receive help in formulating the simulation run parameters by clicking on the parameter for which assistance is desired. The upper right window presents the results of the most recent simulation run. Standard performance statistics (utilization, throughput, waiting time, etc.) are reported for each DFD process and for DFD data stores that are central to the dynamics of the IS design. The developer can receive help in understanding the simulation output, confidence interval generation, and heuristic recom-

mendations regarding potential processing bottlenecks from the simulation results window. A pop-up simulation-output help window is shown in the lower right.

#### 4.1 Automatic Simulation from Data Flow Diagrams

The prototype IS simulator produces simulation results directly from DFDs drawn using an augmented DFD-drawing utility. The mapping from DFDs to simulation is accomplished using the model that each DFD process represents as a server/queue system; that is, each DFD process represents a line of "jobs" and someone/something that accomplishes those jobs, sending them on to whatever process, data store, or external entity is the output of the process. The conventional DFD notation shows the direction and contents of flows through the system but lacks several items relevant to an IS design simulation. Ordinarily, DFDs do not indicate the rate at which jobs are processed, nor do they indicate the amount of time that servers are available to process jobs. The augmentation of DFDs with such dynamics attributes is straightforward. Figure 2 shows the pop-up editing window associated with a DFD process in the augmented DFD-drawing utility. The window in Figure 2 allows the user to specify

1. a designator and label for the process, as normally appears on a DFD,
2. a processing schedule (indicating that the process functions Monday-Friday 8 a.m.-12 noon and 12:30-4:30 p.m., with one server available at those times.) Schedules may include an "and idle" clause (e.g., "12:30-16:30 and idle") to indicate that processing does not conclude until the queue is empty. An end-of-day process can be modeled with clauses such as "P1 off-idle" in lieu of a time range, which indicates that processing begins at the process when processing stops at P1 and continues until the job queue is empty. The time range may also be replaced with "always" to indicate a resource that is continually available;
3. a processing duration, variance, and distribution (an average of 1 minute per job, uniformly distributed between 0.75 and 1.25 minutes);
4. a processing completion rule (which could indicate that the process server should wait for a completed job to be received or completed at a subsequent process before continuing to the next job in the queue);
5. apportionment of output jobs between the possible destinations (e.g., 18% to P2, 10% to E1, and 72% to P3);
6. a textual description of the process.

<sup>1</sup> The implementation of a custom DFD-drawing utility is expedient for creating an integrated environment. Furthermore, a highly controllable DFD editor is desirable for supporting empirical experiments (currently underway) that assess decision support effectiveness of the prototype [13].

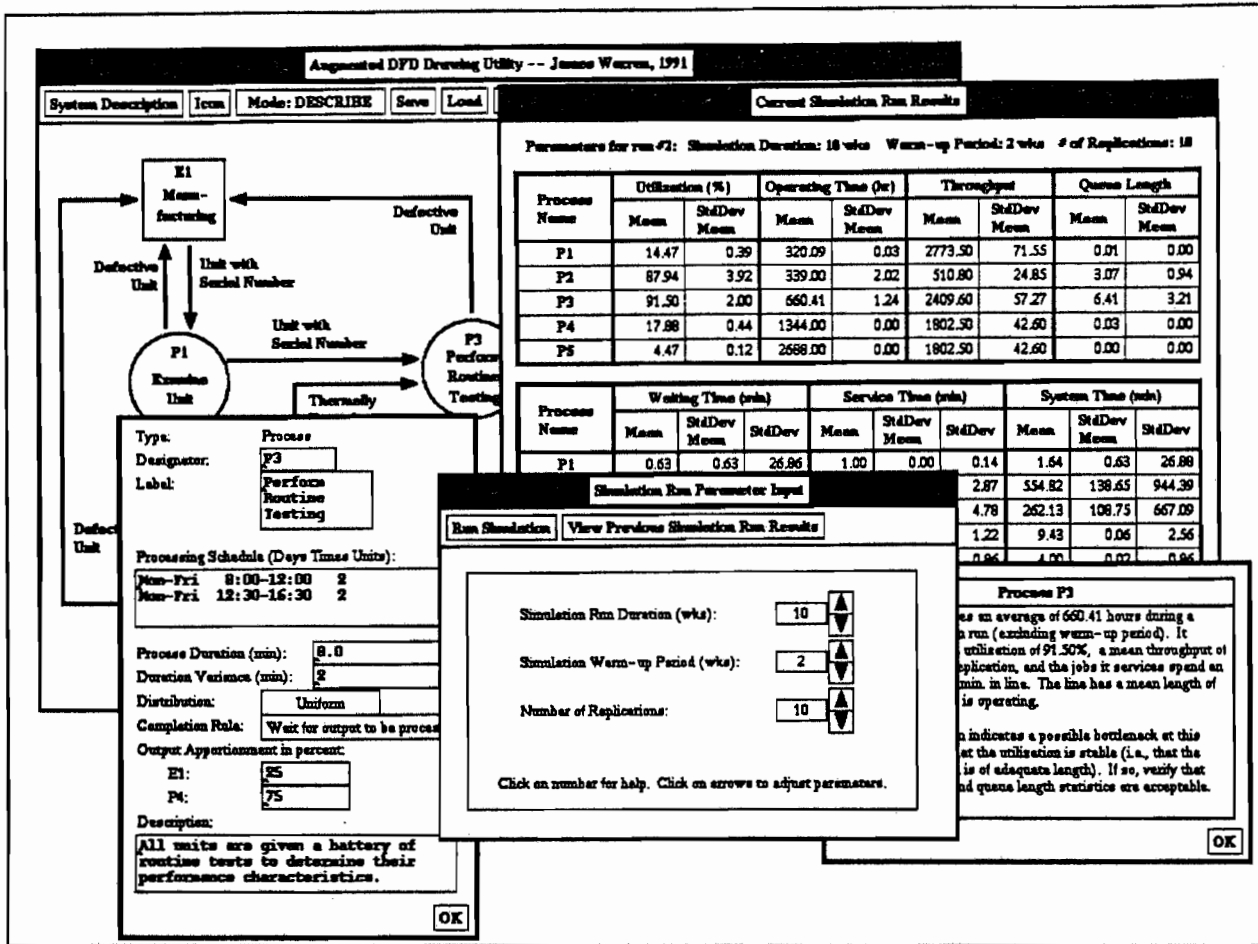


Figure 1. The prototype IS design dynamics simulator runs under X11 on an HP 9000 workstation. The environment presents three main windows: a DFD-drawing utility with pop-up windows to specify attributes of DFD components, a simulation parameter input window with parameter-specification help, and a simulation results window, which explains the output, generates confidence intervals, and provides heuristic warnings of processing bottlenecks.

When the developer selects the "Run Simulation" button on the simulation-run parameter input window, the discrete event-based simulator of the prototype reads the internal representation of the augmented DFD. The simulator maps the DFD to a network of queues, where the DFD attributes define the service rates and distributions for the queuing systems. Note that it would be a minor modification to a middle-level CASE tool such as Index Technologies' EXCELERATOR to allow attachment of attributes to the DFD components, thus allowing the tool to serve as the front end for a simulation environment such as this prototype.

#### 4.4 Simulation Usage Support

The simulation usage support of the prototype is implemented as an online help system rather than in a more obtrusive format. The objective is not to impose a

model on the developer's decision-making style but only to offer assistance in the proper use and interpretation of simulation when the developer requests assistance (by clicking the mouse on the object of interest). Assistance takes the forms of definitions, explanations and calculations, and recommendations.

The developer indicates the need for help by clicking on a parameter in the simulation run parameter input window. The prototype will offer a definition of the indicated parameter and give the developer the option of asking for a recommendation. When the developer requests a recommendation, the prototype analyzes the current simulation results to determine if they are in keeping with its model of proper simulation usage. (If no simulation run has yet been conducted, the prototype offers a heuristic default recommendation and advises that it will be able formulate a more precise recommendation once a simulation has been run.) The simulation

Type:	Process
Designator:	P1
Label:	Examine Unit
Processing Schedule (Days Times Units):	
Mon-Fri	8:00-12:00 1
Mon-Fri	12:30-16:30 1
Process Duration (min):	1.0
Duration Variance (min):	0.25
Distribution:	Uniform
Completion Rule:	Do not wait
Output Apportionment in percent:	
P2:	18
E1:	10
P3:	72
Description:	
Unit is given a cursory examination for defects. 20% of non-defectives are sent for thermal stability tests	
OK	

Figure 2. Pop-up editing window for DFD process attributes.

usage model contains three major principles:

1. The difference between simulation duration and warm-up period should be adequate to allow a sufficiently high number of observations per simulation run to secure normality of the replication means (based on analysis of process throughput).
2. The warm-up period should not end until the running mean utilization of each process approaches a constant level with a given accuracy  $\delta$  [9].
3. Increase in the number of replications or simulation duration should narrow the confidence intervals of output parameters, but lengthening of the simulation duration is the more efficient of the two methods.

The prototype uses the first two of these principles when making recommendations about simulation duration and warm-up period. Either the parameter is indicated to be adequate or larger value is recommended for one or both of the parameters. The third principle is implemented in the prototype by responding to the developer's request for a recommendation on number of replications with an explanation of the relationship between confidence intervals and the simulation run parameters.

The simulation run results window offers output interpretation help in three forms, depending on the area of the simulation output indicated by the

developer:

1. Column headers provide a definition of the indicated output parameter.
2. Row headers provide a summary explanation of all the output shown for the indicated process or data store. A heuristic recommendation is made of the process utilization is sufficiently close to 100% to indicate a possible bottleneck.
3. Cells in the table body provide a confidence interval of the mean of the indicated parameter.

## 5. EXAMPLE USE OF THE PROTOTYPE

We will now illustrate the use of the prototype via application to the quality assurance subsystem of an electronic component manufacturing operation. The IS component of the quality assurance subsystem involves the recording of product testing results into a finished goods inventory data base and collecting thermal stability testing results for a sample of components. Figure 3 shows the DFD of the quality assurance subsystem of an electronic component manufacturing operation.

Manufacturing produces untested units at an average rate of one every seven minutes (following the negative exponential distribution). After a cursory examination of entering units by the shop supervisor (requiring 45-75 seconds per unit), 10% are returned as defective, 18% are selected for thermal stability testing, and the remaining 72% go on to the routine testing stations. Thermal stability testing requires 30-40 minutes; 20% of components will be returned as defective; the remainder go on to routine testing. Two routine testing stations staffed by testers are available. Routine testing requires 6-10 minutes per unit (uniformly distributed across that range) and results in 25% of the units being returned as defective. Testers enter the performance characteristics of components that pass routine testing into the finished goods inventory data base. Data entry, requiring 2.5-6.5 minutes, is performed at one data entry terminal, which pivots between the two stations, and must be performed before testing instruments are cleared to test another unit. Data entry concludes with the printing of a data sheet. After data entry, units are packaged with their data sheet for inventory by the testers, requiring another 2.5-6.5 minutes, and sent to finished goods inventory. Manufacturing and all quality assurance processes operate 8 a.m.-12 noon and 12:30-4:30p.m., Monday-Friday.

The system dynamics are encoded in the attributes of the DFD components using the augmented DFD-drawing utility of the prototype. For example, Figure 4 shows the annotation for external entity E1, manufacturing. The window specifies the times, rates, and

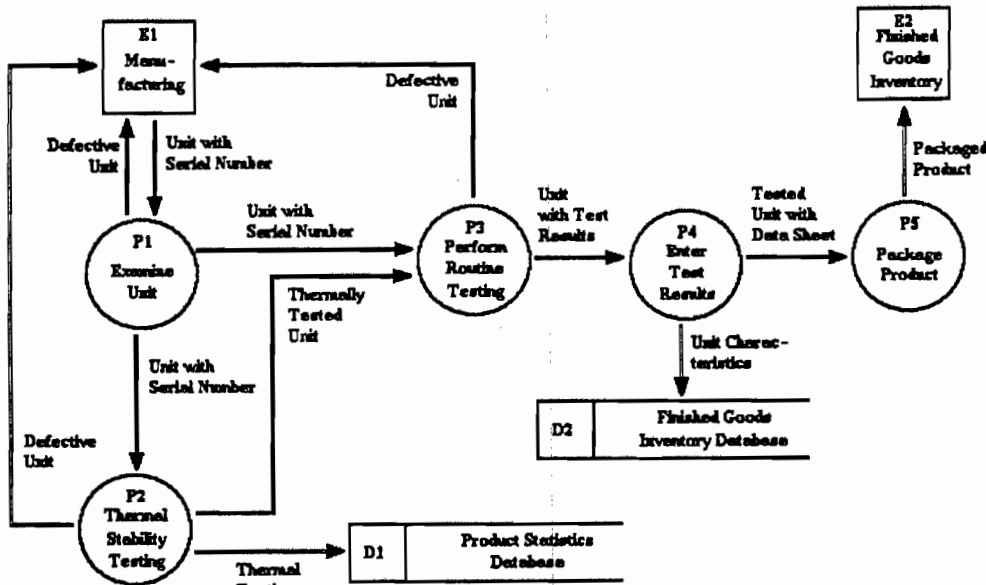


Figure 3. Data flow diagram of the quality assurance subsystem of an electronic component manufacturing operation.

mathematical distributions for jobs arriving at the system from E1 (one unit every seven minutes, distributed exponentially, Monday-Friday during hours of operation). Figure 5 shows the annotation for process P3, "Perform Routine Testing." The processing schedule indicates that two processing units (the testers) are available Monday-Friday 8 a.m.-12 noon and 12:30-4:30 p.m. Fields indicate that the processing rate is 6-10 minutes, distributed uniformly, and that 75% of output jobs proceed to P4 (the "Enter Test Results" process), while the remainder go to E1 (the "Manufacturing" external entity). The completion rule "Wait for Output to be Processed" directs the simulator to wait for a job to be processed at P4 before continuing to process the next job in those cases where output proceeds a process. Processing proceeds immediately to the next job in the 25% of cases where output is to external entity E1.

Figure 6 shows the simulation output results for the quality assurance subsystem with a simulation duration of 10 weeks, warm-up period of 2 weeks, and 10 replications. This simulation requires about 50 seconds of processing time on an HP 9000 series 425 workstation with a 25 MHz 68040 processor and 16 MB RAM. A cursory view of the utilization figures indicates that the system dynamics are not out of control. While the 91% utilization at P3 may be suspiciously close to a bottleneck, the average queue length of around six jobs indicates reasonably smooth operation. Average waiting times of several hours for jobs at the thermal stability and routine testing stations are due to jobs waiting in queue overnight and over the weekend. The

relatively low utilizations and large number of hours of operation for P4 and P5 are due to these processes being modeled as continually available to the testers at P3, who perform the data entry and packaging tasks.

One potentially unsatisfactory aspect of the performance profile seen in Figure 6 is the average waiting time at process P4, that is, the time testers spend waiting for access to a data entry terminal. Clicking on the "Mean Waiting Time" cell of process P4 causes the prototype to pop up the window shown in Figure 7. We are 95% confident that testers wait on average 1.401-1.463 minutes for access to a data entry terminal

Type: External Entity

Designator: E1

Label: Manufacturing

Arrive Schedule (Days Times Inter-arrival-time):

Mon-Fri	8:00-12:00	7
Mon-Fri	12:30-16:30	7

Arrival Distribution: Exponential

Description:

Electronic components are manufactured at a rate of about one every 7 min. throughout the work day.

OK

Figure 4. Annotation for process P3 of the quality assurance subsystem.

Type: Process  
 Designator: P3  
 Label: Perform Routine Testing

Processing Schedule (Days Times Units):  
 Mon-Fri 8:00-12:00 2  
 Mon-Fri 12:30-16:30 2

Process Duration (min): 8.0  
 Duration Variance (min): 2  
 Distribution: Uniform  
 Completion Rule: Wait for output to be processed  
 Output Apportionment in percent:  
 E1: 25  
 P4: 75

Description:  
 All units are given a battery of routine tests to determine their performance characteristics.

OK

Figure 5. Annotation for external entity E1 of the quality assurance subsystem.

before proceeding to packaging. This is completely wasted time, because the testers cannot clear their instruments to test another unit until the results are recorded. Note that the simulation environment merely

**Mean Waiting Time Analysis for Process P4**

Process P4 shows a mean waiting time of 1.43 minutes with a standard deviation of 0.043 minutes. It can be concluded with 95% confidence that the true mean waiting time is between 1.401 minutes and 1.463 minutes.

OK

Figure 7. Confidence interval generation for mean waiting time at process P4 ("Enter Test Results") of the quality assurance subsystem.

describes the scenario; the developer must assess the acceptability of the design dynamics and formulate alternatives to unacceptable outcomes.

In fact, having testers wait an average of 1.4 minutes to perform data entry may be unacceptable. By modifying the annotation to external entity E1 (Figure 4) and rerunning the simulation, one can assess that a bottleneck would develop at P3 if manufacturing upped the rate of production from one unit every seven minutes to one unit every six minutes. The developer investigates two classes of candidate design modifications that offer solutions to this problem:

**Current Simulation Run Results**

Parameters for run #1: Simulation Duration: 10 wks Warm-up Period: 2 wks # of Replications: 10

Process Name	Utilization (%)		Operating Time (hr)		Throughput		Queue Length	
	Mean	StdDev Mean	Mean	StdDev Mean	Mean	StdDev Mean	Mean	StdDev Mean
P1	14.47	0.39	320.09	0.03	2773.50	71.55	0.01	0.00
P2	87.94	3.92	339.00	2.02	510.80	24.85	3.07	0.94
P3	91.50	2.00	660.41	1.24	2409.60	57.27	6.41	3.21
P4	17.88	0.44	1344.00	0.00	1802.50	42.60	0.03	0.00
P5	4.47	0.12	2688.00	0.00	1802.50	42.60	0.00	0.00

Process Name	Waiting Time (min)			Service Time (min)			System Time (min)		
	Mean	StdDev Mean	StdDev	Mean	StdDev Mean	StdDev	Mean	StdDev Mean	StdDev
P1	0.63	0.63	26.96	1.00	0.00	0.14	1.64	0.63	26.88
P2	519.79	138.67	944.40	35.02	0.14	2.87	554.82	138.65	944.39
P3	247.08	108.72	667.01	15.05	0.12	4.78	262.13	108.75	667.09
P4	1.43	0.04	2.25	8.00	0.02	1.22	9.43	0.06	2.56
P5	0.00	0.00	0.00	4.00	0.02	0.86	4.00	0.02	0.86

Click on table header for definitions. Click on cells for specific help.

Figure 6. Simulation output for the quality assurance subsystem.

*Eliminate queuing for data entry.* This could be solved simply by increasing the number of terminals to two. This reduces utilization at P3 to around 85%. Even with one unit coming from manufacturing every six minutes, there would be no bottleneck at P3. Alternatively, the system could be redesigned in an arrangement shown in Figure 8. An in-house data sheet could be filled out by testers, who then proceed immediately to the next unit. In an independent processing sequence, these in-house data sheets could be entered into the finished goods inventory data base and customer data sheets printed. These data sheets are packaged with the tested units. This arrangement may not result in any real saving of labor because testers are likely to spend at least as much time filling out in-house data sheets as they had waiting for the data entry terminal. However, the design promotes specialization of labor in that testers need not perform data entry and packaging tasks.<sup>2</sup>

*Reduce the amount of data entry time.* Halving the data entry time to be uniformly distributed between 1.25 and 2.75 minutes per job lowers the utilization at P3 to about 80% by reducing queuing time to an average of about 0.75 minutes and reducing the data entry time by 2 minutes per job. If entry of routine test results into the finished goods inventory data base were completely automatic, allowing packaging to immedi-

ately follow routine testing, then utilization at P3 would be only 67%. Even if manufacturing upped the rate of production to one unit every 5 minutes, utilization at P3 would be only 90%.<sup>3</sup>

The dynamics of each of these design modifications is easily investigated with the prototype by modifying the annotation to DFD processes (or slightly redrawing the system structure for the alternative shown in Figure 8) and rerunning the simulation. However, implementation to these modifications is a more difficult matter. Can a system with two terminals be delivered within budgetary constraints? Is creating a division of labor between testers and data entry/packaging personnel considered advantageous or disadvantageous by management? Can data entry time be halved? This may be possible by judicious selection of the information that must be entered and/or good human-computer interface design. Can data entry be completely automated? While the benefits of automatic data entry are large, so are the costs. All testing equipment must be interfaced with a computer, which coordinates the testing process and monitors results. The testing software (most likely developed in a conventional programming language such as FORTRAN, PASCAL, or C) will need to interface with the finished goods inventory data base application (more easily developed in a data base 4GL). The hardware requirement will be at least a

<sup>2</sup> Note that labor expenses can be assessed by multiplying hourly personnel costs by the operating time figures provided in the simulation output.

<sup>3</sup> As currently specified, however, a bottleneck would develop at P2 if manufacturing produced one unit every five minutes.

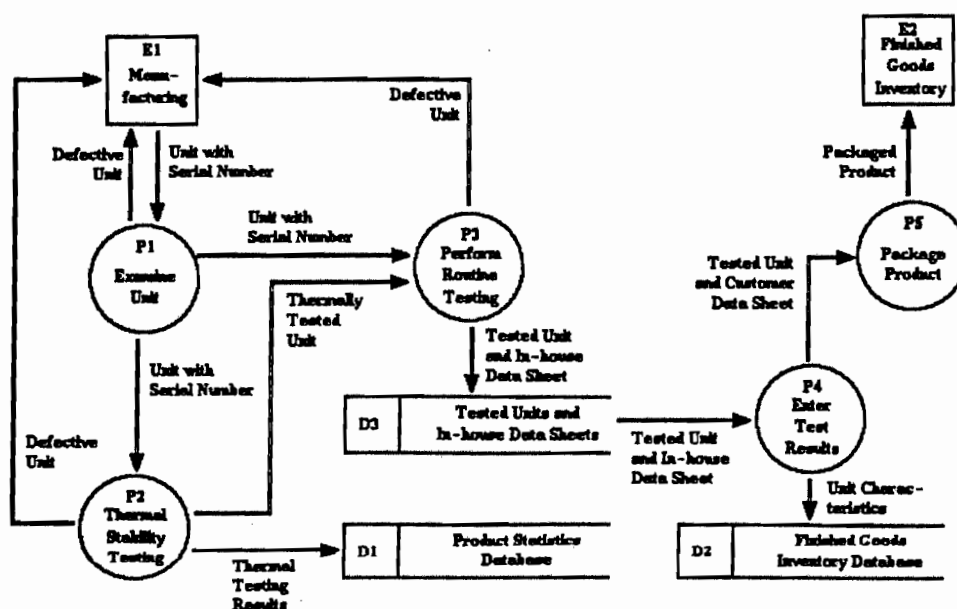


Figure 8. Data flow diagram of an alternative design for the quality assurance subsystem that eliminates queuing of testers at the data entry process by performing data entry and packaging in an independent processing sequence.

microcomputer with networking and instrumentation interface capability, rather than a "dumb" terminal.

## 6. CONCLUSIONS

Direct production of IS design dynamics simulations from annotated DFDs allows the developer to gain the benefits of simulation without sidetracking to code a simulation in a 4GL or to reencode the design structure into a specialized notation. This article has described a prototype simulation environment to support the dynamic modeling of information systems designs from DFD-based system descriptions. In addition to producing stochastic simulations automatically from DFDs, the prototype provides a model for simulation usage and aids in the interpretation of simulation results. The prototype uses a custom DFD-drawing utility to demonstrate the feasibility of the approach; however, the most beneficial configuration would be one in which the DFD-drawing capability and data dictionary of an integrated CASE environment were used to supply the system structure for simulation, as this would allow immediate assessment of the dynamics of proposed designs directly from conventional IS design tools.

The prototype supports the IS developer in determining the most important focus of automation in a proposed IS design. In the example application, it was found that testers wasted time waiting for a data entry terminal. For many designs, improved performance may be accomplished through buying faster computers or DBMS better suited to high-volume processing. For the quality assurance subsystem, solutions included buying another terminal, altering the work flow, redesigning the data entry task, and developing integrated testing and data entry software. The simulation environment allows the developer to quickly quantify the value of design modifications to the system dynamics, but the developer must assess the value of the alternatives within the context of the system requirements, available resources, and organization strategy. At this point, it should be clear how this tool promotes the use of system dynamics simulation as an integral part of the IS development process. Through assessing designs, considering modifications, and assessing the modifications.

The simulation environment answers "what if" questions about IS design performance. However, the prototype must be provided with processing rates based on both machine and human factors for each of the DFD processes. The developer may formulate measures of IS component performance from benchmarking experiments, published reports, systematic observations of task performance, and/or past experience. Once the developer has specified the dynamics of system com-

ponents, the prototype can be asked, "If the components perform like this, what is the overall state of the system dynamics?" This provides evidence for the confirmation (or denial) of the acceptability of the dynamics of the IS design. It also produces a powerful set of dynamics requirements for the IS component designers: If each component performs as well as specified, then the dynamic performance of the IS should be as good as the simulation results indicate. In the difficult, often mind-boggling task of IS development, there is a reassuring feeling about anything as straightforward as "If I do this, then I'll get that."

## ACKNOWLEDGMENTS

We thank Dr. G. C. Canfield, Department of Information Systems, University of Maryland, for helpful suggestions at several stages in the development of the prototype IS simulator.

## REFERENCES

1. W. Inmon, What Price Relational? *Computerworld*, (November 28, 1983).
2. C. Loosely, IBM Database 2 Performance Measurements, *InfoIS* (1985).
3. C. Date, On the performance of relational database systems, in *Relational Database: Selected Readings*, Addison-Wesley, Reading, Massachusetts, (1986).
4. H. Watson and J. Blackstone, *Computer Simulation*, 2nd ed., John Wiley & Sons, New York, 1989.
5. L. Boydston, D. Teichroew, S. Spewak, Y. Yamamoto, and G. Starner, Computer Aided Modeling of Information Systems, in *Proceedings, IEEE COMPSAC80*, 1980, pp. 37-41.
6. D. Teichroew and E. Hershey, PSL/PSA: A Computer-Aided Technique for Structured Documentation and Analysis of Information Processing Systems, *IEEE Trans. Software Eng.* 3, 41-48 (1977).
7. R. Ammar, A Computer Aided Design System to Develop High Performance Software, *J. Syst. Software* 15, 139-147 (1991).
8. C. McClure, The CASE Experience. *Byte* 235-241 (1989).
9. K. Pawlikowski, Steady-State Simulation of Queuing Processes: A Survey of Problems and Solutions, *ACM Comp. Surv.* 22, 123-170 (1990).
10. Y. Park and J. Mellichamp, A Statistical Expert System for Simulation Analysis, in *Proceedings, Summer Computer Simulation Conference*, 1990, pp. 611-616.
11. O. Balci and R. Nance, Simulation Model Development Environments: A Research Prototype, *J. Operat. Res. Soc.* 38, 753-763 (1987).
12. V. Frankel and O. Balci, An On-Line Assistance System for the Simulation Model Development Environment, *Int. J. Man-Machine Stud.* 31, 699-716 (1989).
13. J. Warren, CASE/Simulation Systems, Ph.D. Thesis, University of Maryland, 1992.

# Biographies

**Gen-Huey Chen** received a B.S. in computer science from National Taiwan University, Taiwan, in June 1981, and a M.S. and Ph.D. in computer science from National Tsing Hua University in June 1983, and January 1987, respectively. In February 1987, he joined the faculty of National Taiwan University and is an associate professor in the Department of Computer Science and Information Engineering. His current research interests include protocol engineering, software engineering, design and analysis of algorithms, distributed algorithms, parallel computation, and parallel computer architectures.

**Yung-Chen Hung** received a B.S. in mathematics from Chung Yuan University, Taiwan, in June 1978, and a M.S. in computer science from National Tsing Hua University in June 1986.

From 1986 to 1988, he was a project manager at the Institute for Information Industry. Since September 1988, he has been working toward the Ph.D. degree in computer science at National Taiwan University, Taiwan. His current research interests include parallel and distributed algorithms, software project management, communication network, and protocol engineering.

**Paul Layzell** is currently a senior lecturer in the Department of Computation, UMIST, and a Fellow of the British Computer Society. His teaching and research focuses on methods, techniques, and tools to support software development and evolution. He has worked on several European and UK research projects and is currently project manager of the European Esprit DOCKET project, which is developing a support environment for software system understanding and maintenance.

**A. F. Norcio** is an associate professor in the Department of Information Systems of the University of Maryland, Baltimore County. He is also a computer scientist at the Human-Computer Interaction Laboratory of the Naval Research Laboratory in Washington, DC. His research interests are in the areas of human computer interfaces, software design, and intelligent systems. He has published several dozen papers in a variety of journals and periodicals. He has served on planning committees for many major national and international conferences, and regularly reviews papers for several learned societies journal and professional publishers.

Dr. Norcio is a member of the American Association for Artificial Intelligence, the Association for Computing Machinery, the Human Factors Society, the IEEE Computer Society, and a Senior Member of the IEEE.

**Danny Poo** received his bachelor, master, and doctorate degrees in computation from the University of Manchester Institute of Science and Technology (UMIST) in England in 1983, 1986, and 1988 respectively. His Ph.D. thesis focuses

on the possibility of integrating artificial intelligence technique and object-oriented paradigm for information systems definitions. He is a lecturer in the Department of Information Systems and Computer Science in the National University of Singapore, and is presently involved in the development of an object-oriented modeling approach for business information systems. The approach is based on the use of a rule-based representation of business policies. The objective of the approach is to enhance the visibility of business policies and to provide for ease of change due to changing system requirements. His other research interests include multi-domain expert systems and knowledge and data-based systems.

**Ilié Popescu** received the M.S. and Ph.D. in applied mathematics from the University of Bucharest-Romania, in 1968 and 1976, respectively. Beginning in 1972, he was a research associate and lecturer at the Central Institute for Computer Science, Bucharest. In 1983, he joined the Department of Computer Science, University of Quebec at Hull, Canada, as a regular professor. His current research interests are in the areas of expert systems and knowledge representation. He has authored or coauthored over 30 papers in journals and scientific conferences. Dr. Popescu is a member of several associations: IEEE, ACM, IASTED and CIPS - Canadian Information Processing Society.

**Erhard Rahm** obtained the master's and Ph.D. in computer science from the University of Kaiserslautern, Germany, in 1984 and 1988, respectively. From July 1988 until June 1989, he was a visiting scientist at the IBM T.J. Watson Research Center in Yorktown Heights, NY. Currently, Dr. Rahm is a lecturer in Kaiserslautern and leads a research project on high-performance transaction systems. His current research interests include distributed systems, data base and transaction management, and performance modeling.

**Armin Roeseler** is a member of technical staff in the Computing Technology Department at AT&T Bell Laboratories in Naperville, Illinois. He received a Dipl.-Ing. from the FH-Hannover, Germany, a M.Sc. from the New Jersey Institute of Technology, and a Ph.D. from the Illinois Institute of Technology. His research is in performance analysis and modeling of mainframe computer systems. He has lectured on computer performance metrics and expert systems in the U.S., U.K., Europe, and Australia. Currently, he is adjunct assistant professor of computer science at the Illinois Institute of Technology.

**H. Dieter Rombach** received a B.S. in mathematics and an M.S. in mathematics and computer science from the University of Karlsruhe, Germany, and a Ph.D. in computer science from the University of Kaiserslautern, Germany. He is an assistant professor of computer science at the University

of Maryland. He is also affiliated with the University of Maryland Institute for Advanced Computer Studies (UMIACS) and the Software Engineering Laboratory (SEL), a joint venture between NASA, Goddard Space Flight Center, the University of Maryland and Computer Sciences Corporation. His research interests include software methodologies, process modeling, software reuse, measurement of software processes and products, and automated software engineering environments. He received the *National Science Foundation's* Presidential Investigator Award for his work in software engineering in 1990. He has published extensively on software measurement and its application to maintenance and quality assurance, software process modeling, and software reuse. He has been involved in several projects aimed at introducing measurement into industrial settings. He served as guest editor for the *IEEE Software Magazine* special issue on software quality assurance (September 1987). He is a member of the IEEE Computer Society, the Association for Computing Machinery, and the German Computer Society (GI).

**Jack W. Stott** received a Ph.D. from the University of Arizona where his research included developing the first prototype of PLEXSYS. He has papers published in the *Communications* of the ACM and the *Journal of MIS*. His current research interests are in the modeling of information system dynamics and the visualization of information system requirements.

**Bradford T. Ulery** received a B.A. in mathematics in 1983 from Carleton College and a M.S. in 1985 from UCLA. Currently he is a Ph.D. student in the computer science department at the University of Maryland. He is involved in the measurement and improvement of several maintenance projects at NASA's Software Engineering Laboratory. Other interests include software testing, software fault tolerance, and data bases.

**Jon D. Valett** received a B.S. in computer science from the University of Iowa and a M.S. in computer science from the University of Maryland. Currently he is a software engineer at NASA's Goddard Space Flight Center, where he is responsible for directing and carrying out research in software engineering. He is an active participant in the Software Engineering Laboratory, where his research interests include developing measurement tools for software management, investigating software design measures, studying the software maintenance process, and investigating the impact of CASE tools on software development. Mr. Valett is a member of the IEEE Computer Society.

**James Warren** has just completed his Ph.D. in information systems at the University of Maryland, Baltimore County. Dr. Warren's interests are in the areas of CASE, simulation, and systems design.