Proceedings of the 1991 Winter Simulation Conference
Barry L. Nelson, W. David Kelton, Gordon M. Clark (eds.)

INTEGRATED INTERFACES FOR DECISION-SUPPORT WITH SIMULATION

Philip R. Cohen
Senior Computer Scientist

Artificial Intelligence Center
SRI International

ABSTRACT

A major limitation of graphical user interfaces for simulation is that users such as managers and decision makers need to know too much. We examine the weaknesses inherent in graphical user interfaces to support these users of simulation for short-term situation assessment and scenario evaluation, a style of problem solving characteristic of military and factory command-and-control. Then, we present Shoptalk, a factory command-and-control system with an interface integrating direct manipulation and natural-language processing, and demonstrate how the Shoptalk style of interaction can overcome these limitations.

1 INTRODUCTION

The interface to a simulation-based decision-support system should at a minimum assist the user in iteratively assessing the state of the complex system being studied; expressing, simulating, and comparing scenarios for altering that state; and, assessing the resulting state. Although modern simulation systems employ graphical user interfaces (GUI's), and hence benefit from their advantages, they are hindered by the absence of any provision by GUI's for end users to describe rather than merely select objects. This limitation is particularly severe for nontechnical users of decision-support or command-and-control systems, who would like to solve complex unstructured problems without delving into the intricacies of the underlying computer system. In this paper, we describe a next-generation integrated interface, incorporating graphics and natural-language processing, that overcomes many of the limitations that graphical user interfaces pose for these users of simulation.

1.1 End Users and Their Problems

User interfaces should not be considered in the abstract, but as supporting particular kinds of users in solving specific types of problems. The user we have in mind is the decision maker or manager, who would typically not be the among the developers of the simulation model to be used, and hence would be unlikely to know or want to know details of modeling, simulation, programming or query languages, or database structures. For this kind of end user, the ability to express strategies and questions is vital, as is ease of use in general.

One class of problem such users face is to determine the effects of taking various actions on the short-range behavior of the complex system under study, starting in that system's present state. When the courses of action and the problem solving process itself are routine, a user interface can be tailored to the specific needs at hand. However, decision makers are often called upon to evaluate nonroutine courses of action, or to cope with unforeseen effects of prior courses of action. For such nonstandard cases, a user interface needs to offer the decision maker a conceptually simple but powerful set of tools, emphasizing maximum flexibility and freedom in expressing scenarios and in determining their effects.

For the users in question, unstructured problem solving would cycle through the stages of situation assessment and scenario evaluation. That is, the user would first determine the present state of the complex system, and then decide how to alter it to achieve some desirable end state(s). This process requires that the user be connected to an information system that maintains current and historical information. At the same time, the user needs to have a decision-support system, such as a simulator, to assist in scenario evaluation. Unfortunately, most simulation systems are divorced from the real-world information system (e.g., a factory information system or military command-and-control system) and so they provide different functionality, data structures, and user interfaces. Ideally, the user should have a uniform view of both real and simulated worlds. Were this true, the assessment of the current real-world situation, and the evaluation of the situation that results from a simulation would then use similar techniques. Hereafter, this paper will consider more generally the analysis of situations, but both prospective and retrospective analyses should be understood.

Given this iterative view of problem-solving, we argue that users are impeded by their interface to the information and simulation subsystems. To substantiate this claim, we next discuss strengths and weaknesses of two user interface technologies, direct manipulation and natural language processing.

1.2 User Interfaces

The interface technology of choice in modern simulation systems derives from the direct manipulation style of interaction currently available on workstations and personal computers. With mouse and menu, users can model the system of interest, set parameters, supply attribute values for objects, and animate a simulation run. Although graphics and direct manipulation are effective interface technologies for many classes of problems, they are limited in important ways. Specifically, they provide little support for identifying objects not on the screen, for specifying temporal relations, for identifying and operating on large sets and subsets of entities, and for using the context of interaction. Thus, the user who is trying to solve a problem when he does not know which objects, events, or time periods satisfy his constraints, can only point to entities on the screen, and attempt to determine their relevance. At times, this may be an effective strategy, but in many common circumstances, it can also be tedious or even completely ineffective. Unless a query language is available, and the user is willing and able to learn it, someone else must translate the user's problem into search commands to find the entities of interest. So much for end-user interaction.

On the other hand, the identification of objects, events, and time periods are among the strengths of natural language. That is, English, or any other natural language, provides a set of finely-honed descriptive tools such as the use of noun phrases for identifying objects, verb phrases for identifying events, and tense for describing time periods. Moreover, these capabilities can easily be deployed simultaneously, as in the question "when will the F-15's that are being repaired be operational?"

Because these interface technologies are complementary, they can be merged, affording the advantages of each, and overcoming the limitations each poses for effective problem solving with simulation. Elsewhere, we have described how to merge such interface technologies (Cohen et al., 1989). In the remainder of this paper, we discuss typical problems such users face, and then describe a system, called Shoptalk, that deploys such an integrated interface to make possible new forms of interaction with simulation systems.

2 EXPRESSING SCENARIOS: WHAT-IFS

Proponents of simulation claim that the strengths of such systems lie in the analysis of "what-if" questions. Modern object-oriented simulators (such as SIMKIT[1]) represent an improvement over simulation languages such as GPSS, in that they provide the user with a graphic ability to create a simulation model and to set various input and output parameters. For example, the user could select among various dispatching rules, assign priorities, alter interarrival or service-time distributions, etc.

But, in general, many what-if questions are not easily modeled by merely adjusting parameters. For example, the question "If I give IBM lots top priority, where will the AT&T lots be when the ion implanter goes down for maintenance?" describes an assumption (IBM lots are given top priority), a stopping condition for a simulator (when the ion implanter goes down for preventive maintenance [PM]),[2] and a question (where are the AT&T lots). None of these conditions is easily modeled by a simple setting or tabulating of parameters. For example, the only way to determine which are the IBM lots with present GUI-based simulation systems would be to select each currently displayed lot, examine its attributes to find those destined for IBM, and adjust their priorities. With luck, the system designer would have made the relevant information easily accessible. But, when there are hundreds of entities to be examined (as in many factories), this process would be extremely tedious, and it would be nearly impossible to conduct for lots that have yet to be created (because there would be no icons to select). Rather, the user should be able to characterize the desired objects once, and have the system use the description to find the relevant objects that presently exist, or as they are created. (Of course, specific search and action capabilities could perhaps be added to a sys-

[1] SIMKIT is a trademark of the Intellicorp Corporation.

[2] A scheduling program may already have created a database of future events, in which case this condition would amount to a database retrieval.

tem to handle each of these problems, but at the risk of greatly complicating the interface. In general, if the user needs to write queries in a query language, rules in a rule language, or programs in the host system's programming language, most end users would be lost.)

In summary, our point is that because direct-manipulation GUI's typically afford no means for *describing* entities of interest, decision-makers cannot even *express* the scenarios to be investigated.

2.1 The Analysis of Situations

Typically, the tools provided by simulation systems for analyzing the state of the complex system being studied include graphical displays, as well as the collection and display of statistics computed over the usual suite of variables (e.g., throughput, lateness, down time, etc.) Current interfaces built on object-oriented message-passing graphics will display such charts, both as the simulation is progressing and at its end. In addition, various forms of animation are possible, though the benefits of animation are often debated. It is generally true, however, that a user must know in advance what data he wants to collect before running the simulation. If the user wishes to see unanticipated information, he would typically need to rerun the simulation, with a new set of variables to be tabulated.

However, in the previous section, we argued that many kinds of what-if questions do not merely involve the setting of variables, but rather the satisfaction of complex conditions. Similarly, in determining what has happened, the user may want to do more than tabulate and graph statistics derived from parameter values. More generally, he would like answers to questions. To continue our previous example, the user might want to know where the AT&T lots are, which ones are late, where they waited, for how long they waited, etc. Frequently, the user may not know in advance what questions to ask, but will need to determine the subsequent questions from having seen the answers to prior ones.

Current simulation systems make this style of problem-solving difficult. To support such intelligent exploration of the knowledge base, the simulator needs to record the events that occur and be able to derive new information on demand. At the same time, the interface needs to allow the user to pose arbitrary questions rather than just compute statistics.

In addition to learning about the final state of the simulation, the user may have a bona fide interest in learning about intermediate states. Thus, a simulation system should provide the capability for viewing the state of the system under study at any prior time (Rothenberg, 1986). Though a few systems may provide a "rewind" capability, there are (at least) two reasons why the user interface to that simulation system needs to provide more functionality than merely allowing users to select a prior time. First, the user we are trying to support may simply not know what time is of interest. With systems that allow the user only to scan backwards by dragging a "slider" or typing in a precise time, the user would have to devise a search strategy. A linear search would be particularly poor, while one that merely stepped through the events again, as in a slow-motion replay, would be only marginally better.

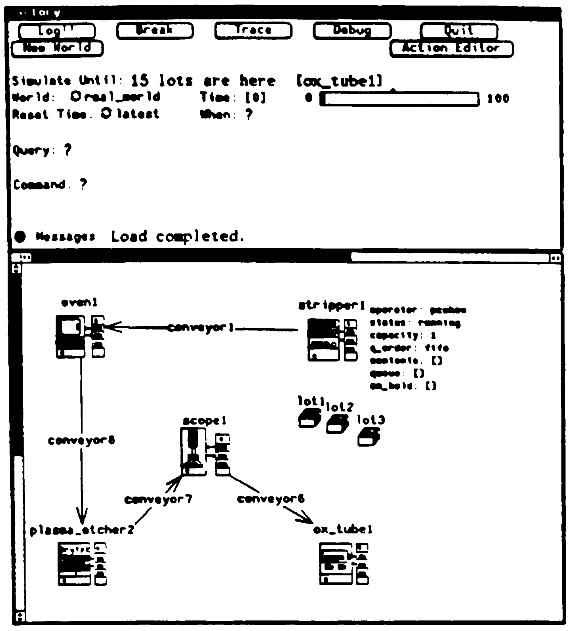
Second, such interface tools for entering times would likely provide only one time per view. But if the decision maker is interested in *all* the times when a certain condition arose, he would have to exhaustively re-apply the temporal search strategy to find the relevant times. Unfortunately, there are simply too many time points from which to pick. The decision maker should be able to describe the condition of interest only once, and rewind the simulation to each of the relevant times.

In summary, we have argued that simulation interfaces that allow decision makers only to perform actions on objects selected from the screen limit the user's ability to express many scenarios and to evaluate their results. Rather, to be useful in solving nonroutine problems, the interface must enable users to describe objects, events, and time periods of interest. The next section describes the Shoptalk system developed at SRI International (SRI) to provide these capabilities.

3 SHOPTALK

Shoptalk is a prototype decision-support system developed at SRI over the past eight years to support such factory situation assessment and scenario evaluation tasks as work-in-progress tracking, quality assurance monitoring, and production scheduling. The current version of Shoptalk demonstrates the application of the technology to semiconductor and circuit-board manufacturing. However, the basic technology is equally applicable to a wide variety of domains. For example, a version of Shoptalk is currently being written for military command-and-control. In this section, we discuss the system, presenting an example of its use for answering what-if questions, and point out features that make it uniquely capable in supporting complex problem-solving.

Shoptalk allows users to query databases on the current state and recent history of the factory and



```
Simulate Until: 15 lots are here [ox_tube1]
World: ○ real_world     Time: [0]
Reset Time: ○ latest    When: ?

Query: ?

Command: ?

● Messages: Load completed.
```

```
operator: [illegible]
status: running
capacity: 1
q_order: fifo
contents: []
queue: []
on_hold: []
```

Figure 1: Factory floor window

to examine alternative factory scenarios by running an event-based discrete event simulator. The simulator is programmed in Prolog, and thus inherits a theorem-prover for determining the truth of complex conditions over a database of facts. Rather than concentrating solely on objects and their properties, as is common in current generation knowledge-based simulation systems, the Shoptalk system is centered around the scheduling, execution, recording, retrieval, and manipulation of events (cf. (Rothenberg, 1991)). For example, in contrast to the usual simulation strategy of determining that an event should occur because its scheduled time has arrived, the Shoptalk simulator takes an action because its invocation condition has become true.

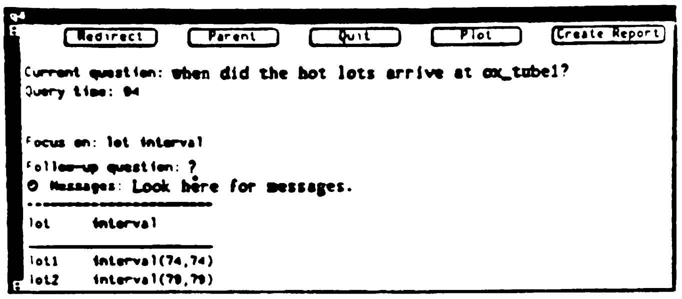
To specify the necessary conditions, the system features an integrated interface that permits intermixing natural-language queries and descriptions with mouse pointing, menu selection, and graphical output. The natural language subsystem is an extension of the Chat system (Warren and Pereira, 1982) based on definite-clause grammars (Pereira and Warren, 1980) written in Prolog. The system parses each natural language phrase into temporally-qualified Prolog expression, whose truth is evaluated by a temporal-constraint satisfier coupled with database retrieval.

The main Shoptalk window contains an iconic depiction of the actual factory floor (see Figure 1). The icons can be manipulated graphically, for example, to show finer grained attributes (see the icon representing the stripper in Figure 1), or to alter parameters, decision-ru'.s, etc. With graphical actions, new objects of various types can be created and the factory layout changed. The user can enter stopping conditions for the simulator from the Simulate Until line of the main window. From the Query line, the user can ask questions about the actual factory's present state and its history. In response to such queries or to graphically invoked actions, data would be retrieved from a Prolog database created by a simulation. For actual operations, such data can be obtained from the main factory computer(s). Fortunately, it is a simple matter to translate Prolog into various query languages, such as the industry-standard SQL language, to retrieve data from relational databases. With this introduction, we first describe how a user can retrieve information from current and historical data using an integrated interface, and then we describe how the same interface can be deployed during scenario evaluation.

3.1 An Interface Integrating Natural Language and Graphics

Pure natural language interfaces for modeling and simulation date back at least to Heidorn's (1973) NLPQ system , which compiled modeling information, given in English statements, into GPSS programs. Shoptalk is more oriented towards using English in the control of simulation and in the analysis of results. The system serves as a testbed for integration of natural language and graphical techniques. The philosophy is to let users employ each technology to its best advantage. Graphical interaction, such as that found on the Apple Macintosh, is effective when the objects of interest are on the screen or are easily found, and when the range of possible actions to be taken is relatively small. Natural language is most appropriate when the user does not know which objects or time periods satisfy his needs, or there are too many objects to conveniently manipulate graphically. In such cases, the user can employ natural-language descriptions to select a relevant object or set of objects. For example, a user can ask a question such as "Where were the defective hot lots when lot 3 was being baked?" without having to know what defects are being discussed, which lots are "hot" (i.e., have high priority), or when lot 3 was being baked.

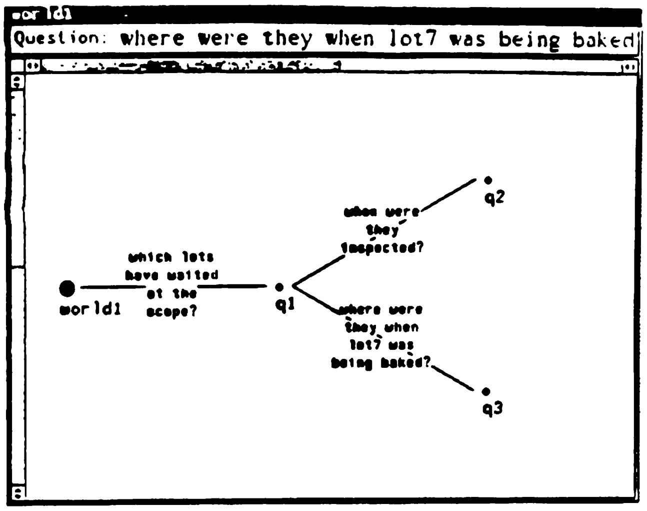
Shoptalk takes advantage of the ability to intermix natural language with graphics to provide users navigational guidance during problem solving, and at the same time avoid some of the more difficult complexities in natural-language discourse processing. Interactive problem solving often starts out with the user's looking at a "big picture," then narrowing the scope of inquiry to particular areas of interest. For a user to be able to engage in an English discourse that nar-



```
Current question: when did the hot lots arrive at ox_tube1?
Query time: 94


Focus on: lot interval
Follow-up question: ?
Messages: Look here for messages.
-------------------------
lot      interval
_________________________
lot1     interval(74,74)
lot2     interval(78,78)
```

Figure 2: "When did the hot lots arrive at ox_tube1?"



Question: where were they when lot7 was being baked

world1
which lots have waited at the scope?
q1
when were they inspected?
q2
where were they when lot7 was being baked?
q3

Figure 3: Context Tree

rows the focus of attention, the interpretation of each utterance will depend on its prior context. Unfortunately, present natural-language processing technology is rather weak in handling the dependence of reference on context in a principled way. Shoptalk incorporates Follow-Up Windows (see Figure 2) as a tool that can be used to maintain context in an interaction, using graphics to overcome this difficult natural-language processing problem. These windows remind the user that follow-up questions can be asked to focus on selected parts of the answer to previous queries. For example, in response to "When did the hot lots arrive at ox_tube1?" the user receives a table indicating that lots 1 and 2 waited there. By selecting those lots, the user indicates that he wants the next follow-up question to focus solely on them. Then, if he asks the follow-up question "When were they inspected?" the user has explicitly restricted the scope of his query to the hot lots that have waited at the scope. The system will respond with a table of answers in a new follow-up window, which will become a daughter in a context tree (see Figure 3). The foci of a given follow-up window constrain the interpretations of references to focused entities of the same type further down in the tree. Thus, further focusing on those lots would specify hot lots that waited at the scope and were inspected.

The user can also select other portions of the context tree to obtain a new context for follow-up questions. With the ability to navigate the context tree freely, the user can maintain several lines of inquiry and follow those that appear most promising.

Another feature of this system is its handling of temporal relations, which are crucial to problem solving with discrete-event systems. Because Shoptalk allows users to employ the full English tense system (Dalrymple, 1988), as our earlier examples showed, the system provides great flexibility for expressing temporal relations. This is one of the areas in which natural language is superior to a purely graphical or query-based interaction. We know of no purely graphical means for posing and answering questions about complex temporal relations, such as the ones implicit in the earlier question. Moreover, current query languages such as SQL do not provide the temporal relationships (e.g., overlapping intervals) needed to answer complex temporally qualified questions.

3.2 Scenario Evaluation

Although decision makers needs to cope with change on an ad hoc basis, their tools are often inadequate. As we have argued, what is needed is software that allows end users to pose and evaluate scenarios flexibly, and to draw comparisons easily. To see how Shoptalk supports such interaction, let us consider a hypothetical scenario.

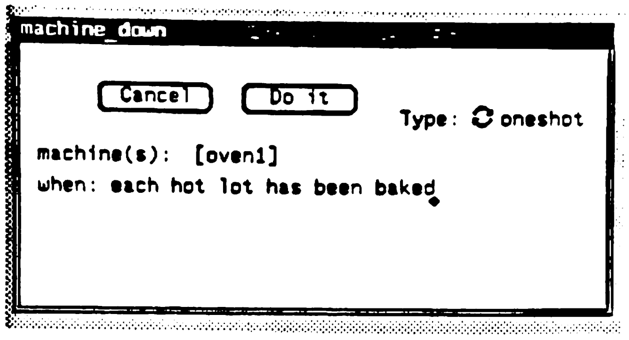
After asking a series of questions, the user determines that the oven temperature is drifting and needs adjustment, a task that takes, for argument's sake, 24 hours. The user decides to compare two operating strategies for the next day: taking the machine down now, or delaying the maintenance action until the high-priority lots have been baked, in the hope that their progress will not be impeded. The user has a complex decision to make, but needs to do so relatively rapidly.

First, the user would push the New World button to get a new partition of the database. This feature allows users to create new hypothetical worlds at any time, providing a tree-structured database, within which various actions or "standing orders" can be taken, either at the start of a simulation, or during the simulation itself. To compare results obtained under various strategies, users can take different actions reflecting those strategies in different worlds. Then, with menus, users can alternate easily between the real world and hypothetical worlds. Because the simulator creates a database of events, and the question-answering interaction retrieves data from it or from the database containing the real-world events, the same user interface can be provided for exploring both real and hypothetical situations.

Currently, the actions that can be invoked include moving lots into a machine, taking machines up or down, adjusting lot priorities, putting lots on hold, and alerting users. The system responds to the selection of an action from a menu by presenting a form, which contains various slots. The user can point at icons and "deposit" them into the slots and/or can type (or speak) English expressions. Shoptalk pårses the content of each slot, determining the objects that are denoted and the conditions under which to execute the action. For example, to specify the second scenario, the user would select the Machine-down action from the action menu, point at the oven, deposit it into the action form, then indicate when the action should take place by filling in the When field with the phrase "the hot lots have been baked" (see Figure 4). During the course of the simulation, and this can be extended to actions that should take place in the real world, when the system can prove that an invocation condition for one of its standing orders holds, it takes the corresponding action. The user does not need to know when that condition will arise, nor precisely which objects satisfy the specified description. Furthermore, the user has neither written pattern-action rules in a formal language, nor asked a programmer to write special-purpose code.

Since the user does not know precisely when the oven will go down, he cannot supply a specific time to bring it back up. But, using the English language capability, to bring the oven on line 24 hours later, the user would simply select the Machine-up form from the action menu, point at the oven, and fill in the When field with the phrase "it has been down for 24 hours."

Finally, the user would decide when the simulation



Type: oneshot

machine(s): [oven1]

when: each hot lot has been baked

Figure 4: Bringing the oven down when the hot lots have been baked

should stop. In most simulators, the stopping condition is specified as a time period (a day, week, etc.). As with other actions the user can specify, the user can state an arbitrary stopping condition in English. Let us assume the user wants to simulate until 15 lots are at the oxidation tube, so he might fill in the Simulate Until field with the sentence "15 lots are here <point to ox_tube1>" (see Figure 1).

Once the future state of the factory has been simulated, the user can then employ the system's query facilities to elicit information of interest. In our case, the user might want to know when the hot lots arrived at the oxidation tube. Figure 2 shows Shoptalk's response to this question.

Asking what-if questions is often one step in a process of making comparisons among different operating regimens. However, the first question that comes to mind after a simulation should not have to be the only question. The user ought to be able to make comparisons of answers to arbitrarily complex questions developed over time. Shoptalk allows for great flexibility in doing so. For comparing results obtained from simulations with differing assumptions, rather than retype a series of old questions, the user can drag the old question into the new world. In the latter case, by dragging nodes (questions and their answers) in the context tree of the old world to that of the new world, the user can re-ask questions that are arbitrarily deeply embedded, and thus have been determined contextually. Consequently, the user need not compare in multiple contexts the answers to just the first question asked, but can explore the space of results derived from one simulation before deciding what to compare. In addition, he can explore alternately the competing scenarios.

Finally, in addition to being able to view the final state of the simulation, the user can rewind the simulation to a time that satisfies some condition by merely specifying that condition in English. For example, the user can find the time when a hot lot was being baked by filling in that phrase in the Reset Time slot of the main window. The system resets the display to that time and constructs a menu for selecting any of the other times at which the condition was true. At any time, the user can create a new hypothetical world and simulate in another direction. This provides a "what-would-have-happened-if" capability in addition to the usual "what-if" analysis.

4 CONCLUDING REMARKS

We have argued that although currently popular graphical user interfaces have provided a new level of ease of use for simulation-based decision-support, they still can prevent the user from expressing scenarios, and hence from solving certain problems. Without the right tools, important questions will not get asked. For example, the scenarios and capabilities discussed above would be difficult if not impossible to obtain with simulation systems employing only graphical interfaces. In this paper, we hope to have shown how an interface taking maximum advantage of both graphics and natural language processing, supported by a logic-based simulation, can make possible new forms of problem solving for decision makers.

5 ACKNOWLEDGMENTS

We would like to acknowledge the contributions of Mary Dalrymple, David Kashtan, Doug Moran, Sharon Oviatt, and Fernando Pereira to the development of the Shoptalk system.

REFERENCES

Cohen, P. R., M. Dalrymple, D. B. Moran, F. C. N. Pereira, J. W. Sullivan, R. A. Gargan, J. L. Schlossberg, and S. W. Tyler 1989. Synergistic use of direct manipulation and natural language. In *Proceedings of CHI'89*, Austin, Texas.

Dalrymple, M. 1988. The interpretation of tense and aspect in English. In *Proceedings of the 26th Annual Meeting of the Association for Computational Linguistics*, Buffalo, New York.

Heidorn G. 1973. An interactive simulation programming system which converses in English. In A. Hoggatt, editor, *Proceedings of the 1973 Winter Simulation Conference*. Association for Computing Machinery, San Francisco, California.

Pereira F. and D. Warren 1980. Definite clause grammars for language analysis–a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, 13:231–278.

Rothenberg J. 1986. Object-oriented simulation: Where do we go from here? In J. R. Wilson, J. O. Henriksen, and S. D. Roberts, editors, *Proceedings of the 1986 Winter Simulation Conference*, pages 464–469, Washington, D.C., December 1986. Association for Computing Machinery.

Rothenberg J. 1991. Knowledge-based simulation at the RAND Corporation. In P. A. Fishwick and R. B. Modjeski, editors, *Knowledge-based Simulation*, Advances in Simulation 4, pages 133–161. Springer-Verlag, New York.

Warren D. and F. Pereira. 1982. An efficient easily adaptable system for interpreting natural language queries. *American Journal of Computational Linguistics*, 8(3):110–123.

AUTHOR BIOGRAPHY

PHILIP R. COHEN is a senior computer scientist in the Artificial Intelligence Center at SRI International. He received his Ph. D. in computer science from the University of Toronto in 1978. His research interests include integrated interfaces, simulation, applications to military and factory command-and-control, and theoretical studies of multiple agents, joint activity, communicative acts, computational linguistics, knowledge representation, and planning.