*Proceedings of the 1986 Winter Simulation Conference*
J. Wilson, J. Henriksen, S. Roberts (eds.)

# SYSTEM ICONIC MODELING FACILITY

Kenneth L. Stanwood
Lani N. Waller
Greg C. Marr
GTE Government Systems Corporation-Western Division
P.O. Box 7188
Mountain View, California 94039

**ABSTRACT**

As government budgets tighten, contractors are increasingly concerned with improving the cost effectiveness of proposed system architectures. One method currently being used to convince customers of the feasibility of a design is the modeling and simulation of the design. The simulation of a proposed design can, however, substantially increase the cost of a proposal, both in dollars and hours.

This paper describes the System Iconic Modeling Facility (SIMF), developed with the goal of minimizing the time required to model and simulate a system, either proposed or existing. The paper emphasizes the properties of SIMF that enable it to achieve this goal.

## 1. INTRODUCTION

SIMF is a graphical interface to Pritsker and Associates' SLAM simulation package. It is written in PASCAL and runs on a Digital Equipment Corporation (DEC) VAX-Station under the MicroVMS operating system. SIMF also runs on Apple Computer Inc's Lisa 2/10 under the workshop operating system.

### 1.1. Point of View

SIMF uses a dataflow representation which allows the construction of models simply by selecting the appropriate SLAM symbols and filling in their attribute fields. It also allows the user to bring the results of running the model through SLAM back to the graphical presentation of his model. The dataflow representation of the model and graphical representation of the statistics allow the SLAM-experienced engineers to rapidly model and simulate systems, greatly increasing productivity.

### 1.2. Why SLAM?

SLAM was chosen as the target simulation environment because it is a well supported, highly flexible, general purpose simulation language. The symbology necessary to implement a graphical interface was already designed and, in fact, sparked the idea to create SIMF. Finally, GTE, the authors' employer, already had a good base of engineers experienced in SLAM.

### 1.3. Features

SIMF contains a number of features that help achieve its goal of maximizing productivity of modeling efforts. Some of these features are:

1. High speed, high resolution graphics
2. Subnets
3. Sophisticated save and restore facilities
4. Parameterized submodels
5. Group cut, copy, and past function
6. Comment node
7. Label node
8. In-field comments
9. Partial error checking at input
10. Interactive compiler
    a. Full field error checking
    b. Compile time correction of errors
    c. Compile time entry of macro parameters
11. Displaying results in context of model.

## 2. HUMAN FACTORS OF GRAPHICS

In keeping with the goals driving the development of SIMF, the major concerns were ease of learning, ease of use, and speed.

Factors leading towards ease of use, for the most part, lead also towards ease of learning. While most ease of use considerations dealt with software aspects of SIMF, there was one major hardware consideration: the method of graphical selection. From the experience of the authors it seemed obvious that SIMF should use a mouse.

SLAM icons are selected by the SIMF user from the icon bar at the top of the screen. The user clicks the mouse when the pointer is over the appropriate icon picture. He then "drags" the icon to wherever he wants. As he does, an actual picture of the icon follows the pointer rather than just a similarly sized box outline. Activities are created by selecting the activity menu selection with the mouse, then selecting the starting point of the activity and "rubber-banding" it to its end point. There is no

need for the user to tediously try to place the end points of the activities on the edges of the icons. As long as the end of the activity is inside the icon it will automatically snap to the edge of the icon.

Similarly, if an icon is positioned on the end of an unconnected activity, the activity will automatically snap to the edge of the icon. Activities may be disconnected from the icons and reconnected elsewhere at the user's desire. If a node is moved, all activities connected to it will automatically reconnect when the icon is released.

When an icon or activity is selected, it is highlighted. Highlighted nodes are displayed inverse video, and highlighted activities are framed. Whenever an icon or activity is selected, its input fields show in the menu area at the bottom of the screen. Fields are selected for input with the mouse and the currently selected field is displayed inverse-video. (See Figure 1.)

Another important feature of SIMF is the ability for the user to select groups of icons and activities. Once they are selected, they may be cut or copied to be pasted elsewhere on the screen or within a subnet.

SIMF does not use color. A monochrome display is much better suited to achieving SIMF's goals. It is easier to move around the icons themselves, to do rubberbanding of activities, and to do highlighting. Most of all though, the monochrome display addresses the third human factor mentioned: speed. No time is wasted figuring out what color to make something, and the hardware need only handle one bit per pixel rather than a byte or word. Another speed consideration was arithmetic. SIMF uses only integer arithmetic to help speed up processing of screen coordinates.

## 3. HIERARCHICAL APPROACH

### 3.1. Description

SIMF provides a hierarchical view of large models, rather than picturing the model as flat and spreading out in all directions. This view is accomplished with the subnet node


Subnet nodes are a very elegant way to show various levels of abstraction within the same model. A large model quite often can start as a number of interconnected subnet nodes. These subnet nodes can then be "expanded" to show the pictorial view of the network they contain. These subnets may themselves contain other subnets providing more detail or space, depending upon their use.

When an entity flowing through the model reaches a subnet node, it continues at the node inside the subnet whose label is the same as the subnet node's label. A return node


is provided for exit from a subnet. Upon

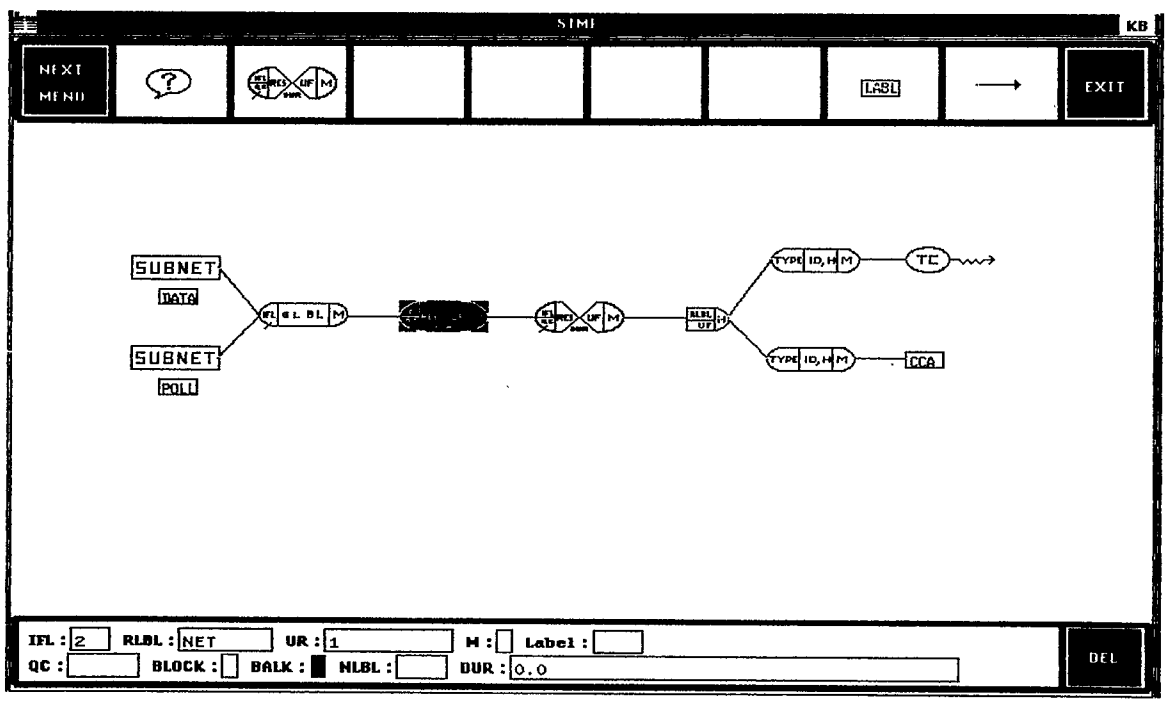


Figure 1: Selected Node Showing Fields

reaching a return node, the entity follows any allowable activities eminating from the subnet node. Entities may, by flowing to a label node




enter a subnet at any node within it that is labeled. Entities may exit a subnet in the same manner, using a label node to branch to any labeled node anywhere in the model. These SIMF "GOTOs," while handy at times, are not highly recommended since they tend to detract from the assistance the pictorial representation gives to the validation process.

### 3.2. Save and Restore Benefits

SIMF's hierarchical approach provided some unexpected abilities connected with the save and restore features of SIMF. Imagine that a user has a model of something such as a computer system, and one of the subnets in the model happens to be something of common interest such as ethernet. He can save that subnet, independent of the rest of that model, for inclusion in future models. Or, imagine that two people have been asked to work together to build a large model that divides nicely into two sections. Once they have agreed upon interface requirements between the two sections, they can each independently model their own section. When done, a model can be created with two properly connected subnet nodes, and the two separate submodels may be restored into the two separate subnets. Both scenarios lead to the reduction in time spent on modeling efforts.

### 3.3. Parameterized Submodels

From the save and restore benefits of subnets was born another idea; parameterized submodels. Parameterized submodels are library subnets created for inclusion as "plug in" components in other models. Many attribute field values in these library models are dependent upon the particular model they are being put into. File numbers, activity durations, and ATRIB numbers for example, are all dependent upon the host model. To aid in the resolution of these dependencies, the submodels can be "parameterized." The developer of the macro encloses any field value or portion thereof in dollar signs to indicate that the value is a parameter rather than an actual value (i.e., ATRIB($data size atrib$). (See Figure 2.)

The user of the macro has two options for giving values to the parameters. When the macro is restored, the user may give values to any parameters he wishes. These changes will be contained in the internal description of his model. Any parameters that he doesn't give values to will be resolved at compile time. The compiler will prompt the user for the values of any macro parameters remaining. These values will show in the SLAM code produced, but will not be put in the internal representation of the model. This allows multiple runs to be made with changes to field values without ever needing to change the model. All the user need to do is recompile. For this reason, parameters are allowed in any model, not just library submodels, thus enabling the user to more easily make multiple runs with different variations in his model.

## 4. ERROR CHECKING

SIMF provides two levels of error checking. The first is a partial error checking of node and activity attribute fields performed when the values are input. Full checking is done for fields that must contain either real or integer constants or labels. The error checking is done on a character by character basis, allowing for backspacing. On other fields, only the first character is checked for membership in the set of possible first characters for the particular field.

The second level of error checking occurs in the compiler. At compile time a complete parsing of all the fields is done. This parsing of the field is complete in the sense that an individual field is checked to ensure that the value it contains is a grammatically correct instance of the possibilities for that field. The relationship between the value and the rest of the model is not checked, but left for SLAM to check (i.e., if the QLAB's field of a select node is "Q1,Q2," SIMF considers it correct even if there are not queue nodes labeled Q1 and Q2).

The compiler's error checking is interactive. Whenever an error is detected, the corresponding icon or activity is highlighted showing the user where the error is. If the error is in an attribute field for an icon or activity, the user is shown the bad value, told which field it is in, and prompted for a new value which is then reparsed. (See Figure 3.) The corrected value appears in the SLAM text produced and is also put into the internal data structure for the model. Other errors, such as disconnected activities, cannot be corrected at compile time and are simply flagged with the corresponding icon or activity highlighted.

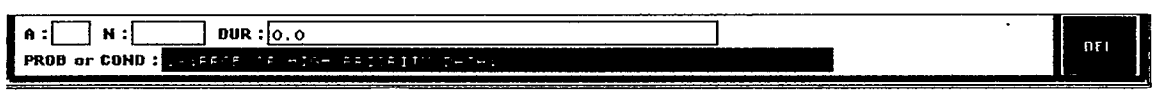


Figure 2: Macro Parameter

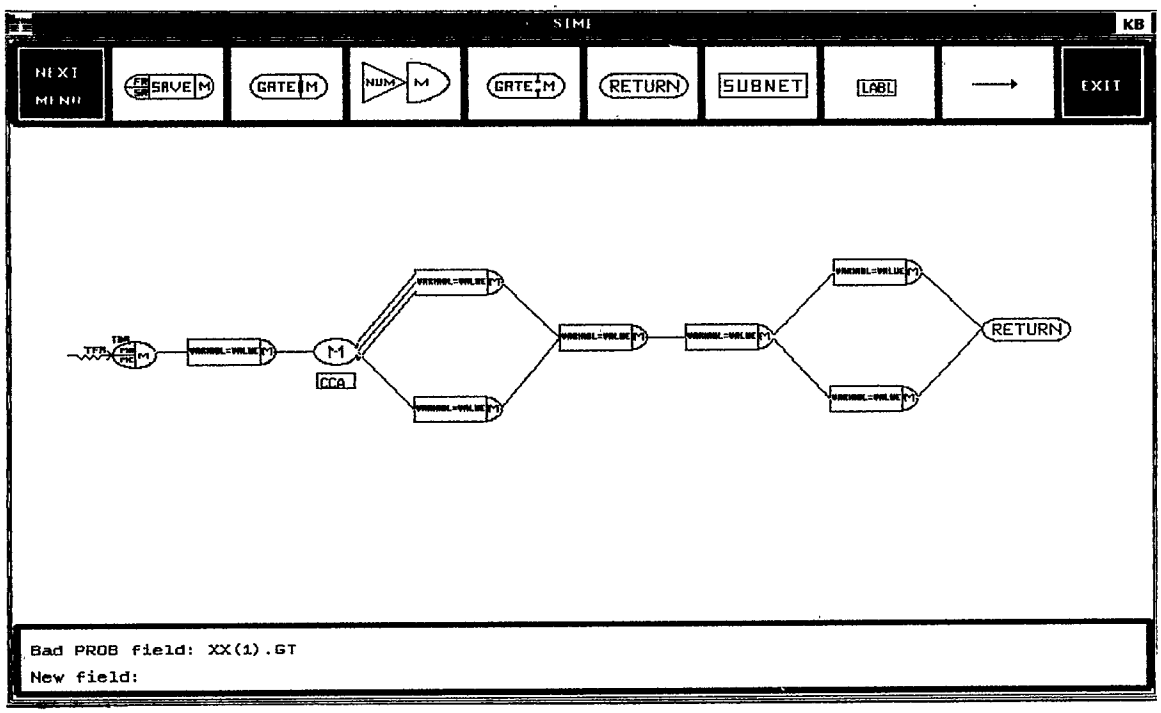


Figure 3: Interactive Error Correction on Compiler

## 5. PROBE

A unique feature of SIMF is the ability to view the results produced by SLAM in the context of the picture of the model. The OTPUT subroutine available for user use in SLAM is used by SIMF (SIMF replaces it by providing a SOTPUT subroutine for user use). This subroutine, written in FORTRAN, is used to collect a condensed version of the statistics automatically generated by SLAM. The statistics returned are:

1. COLCT node statistics
2. File statistics
3. Resource statistics
4. Gate statistics
5. Statistics on numbered activities.

Once a simulation run is made, SIMF may be used in "probe mode" to read in this condensed copy of the results. In probe mode, the user may not create or edit his model, but may view statistics associated with it. As the user selects different nodes and activities with the mouse, instead of the attribute fields showing at the bottom of the screen, any associated results are displayed (i.e., for an await node, file and resource statistics associated with that node will show). This association of results with the physical picture of the model helps speed up the analysis of the results by reducing the effort of determining which results mean what. (See Figure 4.)

## 6. LIMITATIONS

SIMF does not interface to SLAM in its entirety. The match node was not implemented. Also, the continuous portions of SLAM were not dealt with. SIMF has neither the detect node nor control statements pertaining to continuous models. Furthermore, SIMF interfaces to version 2.3 of SLAM at the current time, and does not take advantage of any changes provided in version 3.0 of SLAM.

SIMF has limitations on the size of model it can do. It allows up to 3000 nodes and 3000 activities in 60 subnets. This limitation is necessary in the Apple Lisa version because of memory constraints, but is artificial in the DEC VAXStation version and may be increased.

## AUTHORS' BIOGRAPHIES

KENNETH L. STANWOOD is a Member of Technical Staff of GTE Government Systems. He received a BS in Mathematical Sciences from Oregon State University in 1983. He will be receiving an MS in Computer Science from Stanford University in December 1986.

Mr. Stanwood has four 4 of experience in computer science with his main areas of experience being graphics, signal processing, and the simulation of communications and signal processing systems. Mr. Stanwood has been with GTE since 1983. He has contributed to the design, development, and implementation of graphics interfaces, graphics packages, signal processing

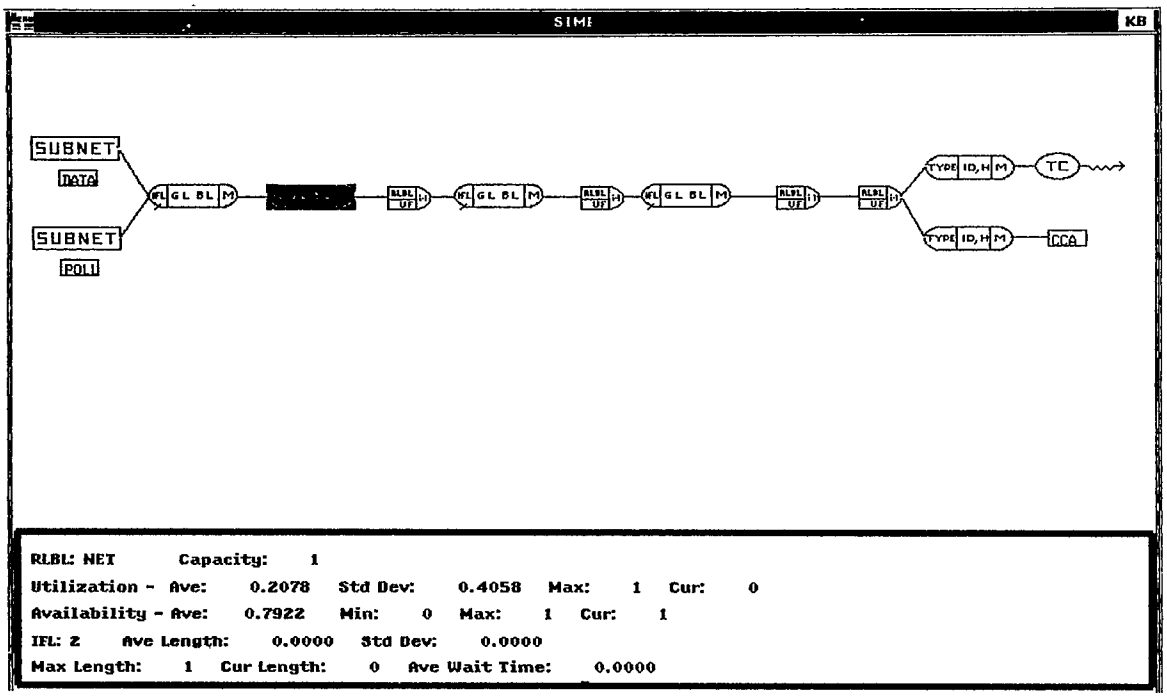


Figure 4: Probe Results

systems, and signal modeling systems. He has also done much work teaching modeling and simulation to system engineers and using modeling and simulation to engineer communications and signal processing systems.

Prior to working for GTE, Mr. Stanwood had a 3 month internship with Tektronix, Inc. He developed graphics software for a CAD/CAM program for developing hybrid and microwave circuits. He also worked on circuit component analysis software while at Tektronix.

Kenneth L. Stanwood
GTE Government Systems Corporation
Western Division
P.O. Box 7188
Mountain View, CA 94039, U.S.A
(415) 966-3580

GREGORY C. MARR is a Senior Member of the Technical Staff of Strategic Research and Engineering. He received a BS in Mathematics from Purdue University in 1968 and a MS in Systems Engineering from Ohio University in 1973. He also did graduate work in Computer Science at Princeton from 1973 to 1975.

Mr. Marr has 18 years experience in computer science with substantial experience in real-time data acquisition and control systems (including satellite communications, and signal collection, processing and analysis applications), and intelligence data-handling systems. He also has experience in real time systems, rapid prototyping, modeling and simulation, data base systems, and telephone switching and networks.

Mr. Marr joined GTE in 1975 and has held both technical and management positions. Since joining GTE Mr. Marr has served as technical manager and/or design task manager for five (5) programs and as project engineer for three (3) internal R&D projects. For the past 2 years, Mr. Marr has worked primarily in the areas of prototyping, modeling and simulation in both a research and development capacity. He has been awarded 2 GTE Superior Performance Awards.

Prior to joining GTE, Mr. Marr was a consultant with TELOS Computing where he was the software systems architect for the SCITIS satellite communications system. Mr. Marr's detailed technical responsibilities included design and implementation of the command and message processing functions, including command encoding, command sequence processing, and transmission control and verification.

Prior to becoming a consultant, Mr. Marr worked for 9 years with Western Electric and Bell Laboratories as a Sr. Information System Specialist. During his employment with Bell, he was responsible for the design and development of software to support Bell System toll switches including TSPS, ETS/4A, and CCIS/STP. In this capacity he served as principal designer and team leader for the development and implementation of both the language and translators (compilers and decompilers) for the MODE and TL1 languages.

Mr. Marr was also designer and developer for the Integrated Data Management System (IDMS) used by the Bell System to manage electronic switching central offices.

Mr. Marr has authored and coauthored publications relating to data base management systems, distributed processing, and language translators.

Mr. Marr has 4 patent applications pending.

Gregory C. Marr
GTE Government Systems Corporation
Western Division
1450 Academy Park Loop, Suite 200
Colorado Springs, CO 80910
(303) 594-7151

LANI NEWMAN WALLER is a Member of Technical Staff of GTE Government Systems. She received a BS in Computer Science from the State University of New York in 1983.

Mrs. Waller has 4 years of experience in computer science with her main areas of experience being signal processing, site automation system, and the simulation of communications and signal processing systems. Mrs. Waller has been with GTE since 1983. She has contributed to the design, development, and implementation of graphics packages, signal processing systems, and site automation systems. She has also done much work teaching modeling and simulation to engineers.

Prior to working for GTE, Mrs. Waller had a 3-month internship with Storage Technology Corporation, Boulder, Colorado. She served as software support to a team of circuit design engineers.

Lani Newman Waller
GTE Government Systems Corporation
Western Division
P.O. Box 7188
Mountain View, CA 94039, U.S.A
(415) 966-3580