SAINT: A COMBINED SIMULATION LANGUAGE FOR MODELING MAN-MACHINE SYSTEMS

Deborah J. Seifert
Department of Operational Planning
AiResearch Manufacturing Company of Arizona

SUMMARY

SAINT is an acronym for: Systems Analysis of Integrated Networks of Tasks. SAINT is a network modeling and simulation technique for design and analysis of complex man-machine systems. SAINT provides the conceptual framework for representing systems that consist of discrete task elements, continuous state variables, and interactions between them. It provides a mechanism for combining human performance models and dynamic system behaviors in a single modeling structure. SAINT facilitates an assessment of the contribution that system components make to overall system performance.

INTRODUCTION

SAINT is a computer simulation tool for modeling and analyzing man-machine systems. While SAINT was designed for modeling manned systems in which human performance is a major concern, it is potentially applicable to to a broad class of problems--those in which discrete and continuous elements are to be portrayed and the behavior exhibits time varying properties. SAINT provides a mechanism for describing these dynamics so analyses can be performed.

SAINT evolved from two separate technologies. Task analysis and the Monte Carlo simulation of operator performance under workload stress as represented by Siegel and Wolf (ref. 1) were the origin for the human factors development. Many of the features eventually incorporated into SAINT were identified as requirements based upon experience in applying this technology. The second origin of SAINT was in the GASP family of simulation techniques (ref. 2). The earliest version of SAINT was an incorporation of the Siegel-Wolf model in a modified P-GERT package (ref. 3). The subsequent evolution of SAINT adapted features of GASP IV and allowed SAINT to become a flexible, sophisticated, combined modeling technique where networks of discrete events could be modeled along with the dynamics of continuous processes.

It is this ability to combine models of dynamics (e.g., aircraft equations of motion) with models of discrete activity sequences (e.g., operator actions) that permits the systems analyst to describe both hardware and human performance in the context of a single model. This affords the system engineer the opportunity to analyze system effectiveness and quantify the relative

contributions of man and machine.

SAINT CONCEPTS

For the discrete event simulation, a graphical-network approach to modeling is taken, whereby a user of SAINT describes the system to be analyzed via a network model and auxiliary descriptions (e.g., equipment and operator performance parameters). A symbol set has been devised for diagraming the discrete task network. The SAINT computer simulation program accepts a description of the network to be simulated and automatically performs an analysis to obtain statistical estimates of system performance. For the continuous process representation, the user is expected to provide FORTRAN statements of the relevant state equations to be solved. Mechanisms are provided for creating an interaction between the discrete and continuous components of the model.

Discrete Task-Oriented Model Component

The discrete task-oriented component of the SAINT model consists of nodes and branches, each node representing a task. Tasks are described by a set of characteristics (e.g., performance time duration, priority, resource requirements). Branches connecting the nodes indicate precedence relations and are used to model the sequencing and looping requirements among the tasks. Complex precedence relations have been designed into SAINT to allow predecessor-successor relationships which are deterministic, probabilistic, and conditional. Resources, either human operators or hardware equipment, perform the tasks in accordance with the network prescribed precedence relations, subject to resource availability. The precedence relations also indicate the flow of information through the network. Information is organized into packets, with each packet containing attributes that characterize the information being processed. The information packet can characterize items flowing through the network, or any other concept related to network flow. When a task is completed, the information packet residing at the task is transmitted along each precedence branch selected. Information attribute values can be assigned or modified at any task in the network and can influence both task performance times and task branching relations.

Resources perform tasks either individually or in groups. Each resource included in a SAINT model is described by a set of attributes. These attributes are also organized into packets, with each packet characterizing a particular resource. Examples of operator attributes include such parameters as level of training, age, height, etc. Machine reliability is an example of an equipment attribute. Resource attributes are used in conjunction with the task descriptions in order to make a general network model resource-specific. The initial values of these user-defined resource attributes are assigned prior to the start of the simulation. The values may be dynamically changed at any task in the network and can be used as parameters in determining both task performance times and precedence relations.

In many instances it may be desirable to specify attributes which are not directly applicable to an information-oriented or resouce-oriented characterization. These attributes are global in nature and do not flow through or move about the network as information and resource packets do. Temperature and time remaining in a mission are examples of model parameters which may be characterized as system attributes. Just as with information and resource attributes, system attributes may influence the task network performance and flow.

Each task in a SAINT network has two requirements which must be satisfied before the task can be performed. First, a specified number of predecessor tasks must be completed before the task is released. Second, once the task has been released, the resources required to perform the task must be available (that is, not be busy performing other tasks). All tasks which have been released (all predecessor requirements have been satisfied) but whose required resources are not available are ranked in a queue according to their priority. Task priority may be assigned at the start of the simulation and may change dynamically as a function of system parameters and contingencies. When the required resources become available with task completions, the tasks in the waiting queue are started. The time to perform a task may be specified as a random variable defined by a probability distribution. SAINT supplies the user with 11 different distributions (Normal, Gamma, Beta, Weibull, etc.)

Frequently the task performance time is also a function of the type of task, the resource or resources performing the task, the status of the system, or the condition of the environment at the time the task is executed. SAINT provides for the specification of factors which influence task performance via user-written moderator functions. It is presumed the modeler can describe (e.g., by least squares techniques) the functional relationshps between a set of conditions and a performance parameter or attribute of interest. For example, one might hypothesize that fatigue affects operator performance such that the average task time increases as a function of mission duration. Research data must be obtained to postulate the functional form of this relationship and fit a curve to these results. This empirically derived relationship can then be implemented in SAINT as a moderator function to determine the possible impact fatigue could have on operator performance. In addition to moderator functions, user-written functions can be developed for specifying attribute assignments. Both types of functions are written in FORTRAN or a FORTRAN-compatible language.

Contingencies, decision making, and emergency conditions can be represented via SAINT's flexible attribute assignment and branching logic. SAINT provides two additional mechanisms for modeling system performance.

The first of these is termed task modification.. This feature enables the user to modify task parameters as a function of ongoing system events. For example, consider a task which may require repetition due to a possibility of failure on the first attempt. The second time the task is performed the performance time may be significantly smaller than the initial execution. SAINT provides for the modification of the task time distribution after the initial attempt. Other task parameters can be modified in a similar fashion.

The second SAINT modeling construct of interest is "clearing". Both tasks and resources can be cleared. "Task" clearing halts a specified task in progress, contingent on the completion of another task. "Resource" clearing halts whatever task the specified resource is performing. Both types of clearing may specify an additional task to be signaled. As an example, consider the simulation of an emergency condition in which all operators must stop their ongoing activites to assist in the emergency operations. This situation is best modeled in SAINT with resource clearing. The onset of the unexpected event would "free-up" (clear) the operators. Concurrently, emergency handling tasks would be signaled for initiation (and release if all other precedents were satisfied). Task and resource clearing provide dynamic realism in man-machine simulation modeling. The network symbol used to diagram a task in a SAINT model is illustrated in Figure 1. The input side of the node reflects the precedence requirements for releasing a task. The number of requirements for releasing a task the first time is on the top (PR1) and the number of requirements for releasing a task on subsequent times is on the bottom (PR2).

The center portion of the task symbol contains all task description information, such as performance time characteristics, statistics to be collected, and attributes to be assigned. It is subdivided into rows, with each row containing a specific type of descriptive information about the task. Further, each row is divided into two parts. The left-hand part contains the task description code. It is used to identify the type of information that appears in the right-hand part of the row, and can be any of the 17 available codes shown in Table I.

The LABL permits an eight character identifier to be associated with this node to depict the nature of the task/activity represented. The TIME parameters indicate the distribution type and parameter values for the characterization of task duration. If activity times are known to be a function of specifiable factors (e.g., task, system, or information attributes), a moderator function (MODF) may be employed (as a FORTRAN subroutine) to generate the activity duration instead of generating a time value by Monte Carlo methods. If Monte Carlo methods are employed (via TIME specification), a modification can be effected during model execution by using the DMOD feature to identify an alternate distribution and/or parameter set when specified event conditions prevail. RESR may be used to specify the type and quantity of resources and whether multiple resources imply substitution ("or") rather than conjoint ("and") requirements. If priority (PRTY) is a concern, it can be specified a priori and subsequently manipulated dynamically during model execution. Since information packets can arrive at a task from several sources, but only one will exit, it is necessary to specify which incoming packet will be passed along, INCM. The default condition for processing information packets is to simply pass the last one arriving at the node. If different predecessor completions are required in order for the task to be released, the DIFF option must be specified. Otherwise the multiple occurrence of any predecessor may cause the task to be prematurely released. When two or more tasks have identical completion times, it is necessary to specify which will take precedence (PREC) over the others. User-defined task characteristics (UTCH) permit the user to specify additional attributes of a task (e.g., difficulty, complexity, etc.), and these attributes can be modified upon task execution. Information, re-

source, and system attributes can be assigned or updated (ATAS) upon task release, start, or completion as required. The statistics to be collected (STAT) are described in subsequent discussion. A particular task can be used to mark the start point (MARK) for timing how long it takes to traverse a path to some other task of interest. The MARK feature allows elapsed time computations within the network (e.g., time between events). Task and resource clearing operations are established by specifying the appropriate parameters associated with the TCLR and RCLR mnemonics. Upon completion of a task, SWIT allows a switch or flag to be set for subsequent examination in the continuous state variable component of the model. The REGL mnemonic is used as a device for regulating values employed in the continuous process model, where a task is permitted to alter a state variable, for example.

By selectively using these description codes, only the information necessary to describe a task need be shown on the task symbol. In this manner, any or all of the task description codes can be specified for a particular task. If more than the four rows provided are required for a complete description, the user simply adds the necessary number of additional rows to the bottom of the task description portion of the task symbol.

The output side of the node contains the task number (TSK). In addition, the shape of the output side indicates the branching operation to be performed upon task completion. It specifies the process to be employed in selecting the successor tasks whose precedence requirements should be reduced by one. The four branching types included in SAINT are deterministic, probabilistic, conditional take-first, and conditional take-all. Their shapes are depicted in Figure 2.

When deterministic branching is specified, the number of requirements for all successor tasks is reduced by one. For probabilistic branching, each branch emanating from the task has an associated probability of selection. These probabilities may be specified directly or obtained from information, operator or system attributes. Only a single successor task is selected. For conditional take-first branching, each branch has an associated condition, and the branches are ordered. Each condition is tested in the prescribed order, and the first branch whose condition is satisfied is selected. Conditional take-all branching operates in the same manner, but selects all branches whose conditions are satisfied. Conditions may be based on task completions, simulated time, or attribute values.

The above discussion only included the basic task node symbology. Additional symbolism is available for task modification, task signaling as a result of task or resource clearing, task signaling resulting from a threshold crossing, and state variable monitors (refs. 4 and 5).

Continuous State Variable Model Component

The second component of a SAINT model is the state variable description. The SAINT user defines these state variables by writing the algebraic, difference, or differential equations that govern their time-dependent behavior. The use of state variables in SAINT is optional.

The SAINT user writes the state variable equations in a FORTRAN subroutine (subroutine STATE). State variables represented by algebraic or difference equations are defined in subroutine STATE as SS(•) variables. Those represented by differential equations are written in terms of DD(•) variables. SAINT employs a Runge-Kutta-England (RKE) numerical algorithm to integrate the equations of subroutine STATE written in terms of the DD(•) variables. The RKE algorithm obtains a solution to a set of simultaneous first order ordinary differential equations. Higher order differential equations can be modeled by placing the equations in canonical form. Subroutine STATE can be used to model state variables using a combination of DD(•) and SS(•) variables.

In SAINT, simulated time is advanced in accordance with the type of system being modeled. If no state variables are included, simulated time is advanced from one task completion to the next. When state variables are included in the model, time is also incremented in steps between scheduled task completions for the purpose of updating the values of the state variables. The step size is a function of user-specified accuracy requirements.

Discrete and Continuous Component Interactions

The interactions between tasks and state variables are initiated either by tasks being completed or by state variables crossing specified threshold values. Upon the completion of a task, state variables may be discretely regulated by increasing or decreasing their values. In addition, task completions can change the values of logical variables which can be used to alter state variable equation forms or the network structure. In this manner the discrete task-oriented component of the model affects the continuous state variable component.

Threshold crossings by state variables can signal or initiate tasks. Thus the values of state variables can influence task performance characteristics and precedence relations. Threshold crossings can also change the values of logical variables which, in turn, can be used to alter equation forms or change task precedence.

As an example of discrete and continuous component interactions, consider a system in which a pilot must keep the aircraft altitude within specified constraints. The pilot's inputs might be modeled as discrete tasks and the aircraft dynamics as continuous state variables. When the altitude state variable crosses the allowable threshold value, the corresponding discrete pilot makes the appropriate input and regulates the state variable(s) which determine altitude. Thus, through this component interaction, the aircraft altitude is brought back within acceptable limits.

STATISTICAL OUTPUT

Once the model has been built, the modeler can impose a data collection structure to obtain information about his description of the system as it is exercised. A variety of data can be obtained; these fall into four major

categories. The first type of output is a statistical description of the execution of specific nodes or collections of nodes. There are sixteen possible combinations of interval and task completion statistics that one can collect using the built-in features of SAINT. Since users can create their own functions for updating attributes and for moderating network parameters, it has been necessary to allow the user to collect his own statistics on those parts of the model which cannot be predefined because the user creates them himself. SAINT supplies statistical subroutines for collecting data on user-supplied parts of the model. Tabular summaries of the computed descriptive statistics can then be generated to portray the results of a single iteration, a set of iterations, or a series of iterated runs showing the trends induced by some systematic variation of run conditions.

A second type of output which SAINT provides is resource utilization statistics. Information on the busy/idle status for both the human resources as well as the equipment resources is automatically presented at the completion of each simulation run. These statistics can be employed in evaluating workload and system capacity issues.

The third type of output is a graphic portrayal of the probability and cumulative density functions for a distributed variable. These histograms provide a quick look at the shape of the data. An experienced user can store the actual values on an external device; later, the data can be fed to a plotting package for reproducible drawings.

Time traces of the state variables are a fourth type of output. Up to 10 variables can be plotted on the same graph with user specified scale factors and plotting symbols. Multiple graphs can be generated. Tabled values of the variables can also be obtained. The tabulated plot provided by SAINT equips the user to quickly examine the results of his simulation run.

THE SAINT PROGRAM

Development of the SAINT simulation package has been completed and is fully documented (refs. 4, 5, 6, and 7). SAINT was developed in ANSI standard FORTRAN and, consequently, is machine-independent. The user, however, must supply his own system-specific random number generator. The task network data is punched on cards in free-form. SAINT includes an extensive input error-checking feature to assist users in debugging their models. For production runs, users can select a more efficient non-error-checking version of SAINT. A separate FORTRAN program has been devised to create a source module with the COMMON blocks sized to the problem being run. SAINT also includes provisions that allow formatting model outputs so they can be processed by available statistical analysis packages (ref. 7).

APPLICATIONS OF SAINT

SAINT has been used to analyze a wide variety of man-machine systems. It

is gaining a wide and enthusiastic acceptance by systems modelers and analysts of many disciplines. The following is a list of completed or ongoing modeling and simulation efforts involving the use of SAINT: SAINT has been used by the Aerospace Medical Research Laboratory (AMRL) to evaluate alternatives for a Remotely Piloted Vehicle/Drone Control Facility (RPV/DCF) in which operators monitor and control the flight of RPV's through the use of visual (CRT) displays (ref. 8). SAINT was used, also, to provide flight control performance predictions for the Digital Avionics Information System (DAIS) cockpit configuration in which dedicated instruments, displays, and subsystem status displays have been replaced with interactive multipurpose displays and multifunction keyboard switching. A first model of this system employed discrete task networks to represent the pilot's activities and continuous state equations to represent the vehicle dynamics (ref. 9). More recently, a model of DAIS has been developed in which the pilot's discrete information storage and retrieval activities were modeled by tasks; however, the pilot's flight control was represented by a variation of the Optimal Control Model developed by Bolt, Beranek and Newman. In this combined discrete/continuous model of the human operator the pilot operates in a so-called "open loop" preprogrammed fashion between flight control variable sampling (ref. 10). SAINT is currently being used by AMRL to provide cost trade-off design analyses of proposed alternative configurations for the UPD-X All Weather Wide Area Surveillance ground exploitation station. AMRL plans to utilize SAINT to analyze design proposals in a B-52 strategic navigation system involving complex crew activities and task management (ref. 11). SAINT has been used by the Air Force Human Resources Laboratory to explore the feasibility of employing computer simulation for evaluating human effects on nuclear systems safety in a missile loading operation (ref. 12). SAINT was employed by Air Force Weapons Laboratory to examine workload sharing and nuclear radiation effects on pilot performance in an air-to-air refueling mission (ref. 13). Purdue University researchers utilized SAINT to investigate the effect of higher degrees of automation, different capacities of process limiting operations, and alternative task allocations on the operator's idle times in a hot strip mill (ref. 14). SAINT has been used by the U. S. Department of Commerce Office of Telecommunications to analyze communication frequency utilization in a railroad switching yard. SAINT has been used by New Mexico State to compare theoretical human performance predictions with empirically derived performance data (ref. 15). SAINT is being employed by Pritsker and Associates in support of the Army Research Institute to analyze human system performance in an AN/TSQ-73 guided missile air defense system operation (ref. 16). SAINT is also being utilized by several universities both in the classroom and for research activities. Among these are Purdue, Iowa State, North Carolina State, Ohio State, and Arizona State.

REFERENCES

1. Siegel, Arthur I.; and Wolf, J. Jay: Man-Machine Simulation Models: Performance and Psychosocial Interaction. New York, John Wiley & Sons, Inc., 1967.

2. Pritsker, A. A. B.: The GASP IV Simulation Language. New York, John Wiley & Sons, 1974.

3. Pritsker, A. A. B.; Wortman, D. B.; Seum, C. S.; Chubb, G. P.; and Seifert, D. J.: SAINT: Volume I, Systems Analysis of Integrated Networks of Tasks. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-78-126, 1974.

4. Wortman, D. B.; Duket, S. D.; Seifert, D. J.; Hann, R. L.; and Chubb, G. P.: Simulation Using SAINT: A User-Oriented Instruction Manual. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-77-61, 1978.

5. Wortman, D. B.; Duket, S. D.; Seifert, D. J.; Hann, R. L.; and Chubb, G. P.: The SAINT User's Manual. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-77-62, 1978.

6. Duket, S. D.; Wortman, D. B.; Seifert, D. J.; Hann, R. L.; and Chubb, G. P.: Documentation for the SAINT Simulation Program. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-77-63, 1978.

7. Duket, S. D.; Wortman, D. B.; Seifert, D. J.; Hann, R. L.; and Chubb, G. P.: An Example to Illustrate the Use of SPSS for the Analysis of a SAINT Model. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-77-64, 1978.

8. Wortman, D. B.; Duket, S. D.; and Seifert, D. J.: SAINT Simulation of a Remotely Piloted Vehicle/Drone Control Facility: Model Development and Analysis. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-75-118, 1976.

9. Kuperman, G. G.; and Seifert, D. J.: Development of a Computer Simulation Model for Evaluating DAIS Concepts. Proceedings: Human Factors Society 19th Annual Meeting, Dallas, Texas, Oct. 1975, pp. 347-353.

10. Seifert, D. J.: Combined Discrete Network — Continuous Control Modeling of Man-Machine Systems, Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-79-34, March 1979.

11. Chubb, Gerald P.; and Berisford, Kathleen M.: Manned System Modeling, SAINT Applied to Strategic Navigation. Record of Proceedings, Tenth Annual Simulation Symposium, 1977, pp. 321-352.

12. Askren, W. B.; Campbell, W. B.; Seifert, D. J.; Hall, T.; Johnson, R. C.; and Sulzen, R. H.: Feasibility of a Computer Simulation Method for Evaluating Human Effects on Nuclear Systems Safety. Air Force Human Resources Laboratory, Brooks Air Force Base, AFHRL-TR-76-18, 1976.

13. Wortman, David B.; Sigal, C. Elliot; Pritsker, A. Alan B.; and Seifert, Deborah J.: New SAINT Concepts and the SAINT II Simulation Program. Aerospace Medical Research Laboratory, Wright-Patterson Air Force Base, Ohio, AMRL-TR-74-119, Section VI, 1975.

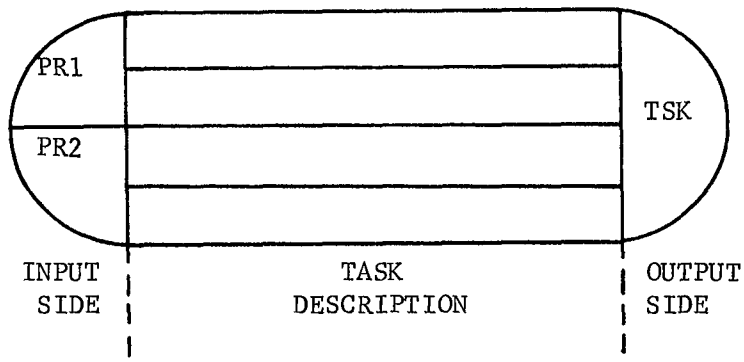
14. Maltas, Kenneth L.; and Buck, James R.: Simulation of a Large Man/Machine Process Control System in the Steel Industry. Proceedings: Human Factors Society 19th Annual Meeting, Dallas, Texas, Oct. 1975, pp. 193-205.

15. Teichner, Warren H.: Human Performance Simulation. Annual Report, AFOSR Contract No. F44620-76-C-0013, Bolling Air Force Base, Washington, D. C., 1976.

16. Wortman, D. B.; Hixson, A. F.; and Jorgensen, C. C.: A SAINT Model of the AN/TSQ-73 Guided Missile Air Defense System. Proceedings of the 1978 Winter Simulation Conference, December, 1978, pp. 879-888.

TABLE I
TASK DESCRIPTION CODES

LABL	task label
TIME	performance time characteristics
MODF	moderator functions
DMOD	distribution modification
RESR	resource requirements
PRTY	priority
INCM	information choice mode
DIFF	different predecessor option
PREC	completion precedence
UTCH	user-defined task characteristics
ATAS	attribute assignments
STAT	statistics
MARK	mark information
TCLR	task clearing
RCLR	resource clearing
SWIT	switching
REGL	regulation



PR1 NUMBER OF PREDECESSOR COMPLETIONS REQUIRED FOR FIRST RELEASE OF THE TASK

PR2 NUMBER OF PREDECESSOR COMPLETIONS REQUIRED FOR SUBSEQUENT RELEASE OF THE TASK

TSK TASK NUMBER

Figure 1.- Task symbol.

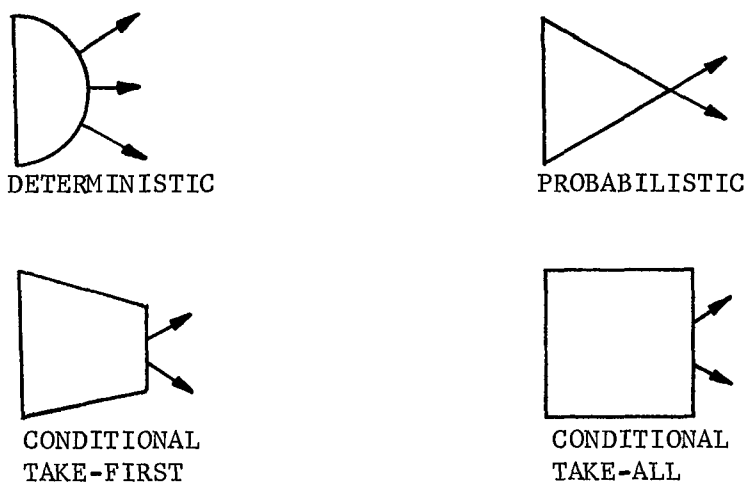


Figure 2.- Task branching symbolism.