1-1-2005

A Comparison of SAINT with IMPRINT and Micro Saint Sharp

Gerald P. Chubb

A COMPARISON OF SAINT WITH IMPRINT AND MICRO SAINT SHARP

Gerald P. Chubb
The Ohio State University
Columbus, Ohio

SAINT was a hybrid modeling and simulation language developed in FORTRAN for main frame computers that allowed simulation of human activities in the context of system operation. MicroSaint was initially developed in the C language, specifically for Personal Computers (PCs), mimicking much but not all of what was in the original FORTRAN version. IMPRINT Version 7 uses Micro Saint IV as its underlying computational engine. MicroSaint Sharp is based on the C# programming language and will be the computational engine underlying IMPRINT Version 8. Representational capabilities of these various modeling techniques are compared to illustrate what improvements have been made and what has been abandoned in the progressive development of these analysis tools.

Introduction

SAINT is an acronym standing for Systems Analysis of Integrated Networks of Tasks. As a modeling tool, it uses a general activity network to represent procedural and decision making tasks, including parallel activity chains by one or more operators. The associated software executes a Monte Carlo simulation of the activity network, generating statistics on activity duration, time of task sequence completion, number of task repetitions, and other descriptive measures. There are numerous similar diagramming techniques, such as: DeMarco Data Flow Diagrams (Yourdan and Constantine, 1979), Petri Nets (e.g., Desrochers and Al-Jaar, 1995), and PERT charts (Moder, et al., 1983). While PERT uses an activity-on-branch representation, SAINT uses an activity-on-node representation.

Background

The original impetus for developing SAINT was the Siegel-Wolf two-man operator simulation model (Siegel and Wolf, 1969), used in a study of F-106 nuclear vulnerability / survivability (Chubb, 1971). Task times were assumed to be normally distributed with some specified probability of success. Failed tasks led to repetition of the task. Average and standard deviations of the nominal task durations were adjusted to reflect the impact of time stress, as determined from time available versus time required. It was recognized that: 1) engineers were reluctant to use a model developed by psychologists, 2) there were other distributions of task times that might better represent certain activities, 3) the branching structure logic was simplistic, and 4) there was no representation of system dynamics that might drive human performance. To be effective, it was believed the best approach was to use simulation technology that engineers were taught to use and then incorporate human factors considerations into that technology. Such was the goal for SAINT.

SAINT Development

SAINT was developed using elements of GERT (Pritsker and Happ, 1966 and Pritsker and Whitehouse, 1966), a FORTRAN simulation language used by industrial engineers to model discrete systems, later adding elements from GASP IV (Pritsker and Hurst, 1973) that also allowed representation of continuous system dynamics (Cellier,1982). For a more detailed overview of the initial SAINT modeling and simulation capabilities, a list of preliminary applications, and references to documentation see Seifert and Chubb (1978).

Subsequent Applications and Developments included modeling of the B-1A Electronically Agile Radar (EAR) to determining the characteristics of time-sharing between forward looking terrain tracing and horizontal ground mapping modes. Additional branching logic and other modeling improvements were also made under the Cockpit Automation Technologies (CAT) program (Hoyland, et al., 1988). SADT (Marca and McGowan, 1988) was also shown to provide a good top-down, front-end analysis technique consistent with later developing the SAINT activity networks (Chubb, 1989).

Deficiencies and Shortcomings of SAINT included a lack of graphics capability to represent network models, complicated symbology for network diagramming, particularly the types of branching logic, and the general lack of technical support. While the source code was delivered with no restrictions on data rights (therefore available in the ‘public domain’ and releasable to any requester), there were no formal provisions for giving users any technical support if they encountered difficulties in their use of SAINT. SAINT was designed as a batch program for a large, main frame computer, and all data was in alphanumeric form using punched cards. Micro Saint (Laughery, 1985) changed that, substantially.

Micro Saint Development

Micro Saint was developed by Micro Analysis and Design (MA&D) as part of the Army's Chemical Defense program and made four major improvements: 1) it was hosted on personal computers (PCs), b) it included graphical representations of the network model, 3) it simplified the representation of branching logic, and 4) data entry was easier and less error prone. Micro Saint was initially programmed in the C programming language, which permitted capabilities not readily available to FORTRAN programmers. Network graphics were animated to indicate which task(s) were being executed as the sequence unfolded. In addition to helping users better understand the task-flow, it also helped 'debug' incorrectly implemented models. There was no attempt to include all the distribution types or types of branching logic found in SAINT, nor did Micro Saint include SAINT's continuous-time modeling of system dynamics.

The most recent version of Micro Saint is based on C#, not C or C++. This new language offers more programming power and additional capabilities (Bloechle and Schunk, 2003). The ability to build enhanced animation of models has also been added, as well as permitting the development of better web-based applications. However, developing advanced animations may, by itself, take as long as developing the model and Micro Saint simulation. It has a distinct cosmetic advantage in promoting a model and its use, but does little technically – the underlying model is the same. An add-on optimization package is also available for analyzing the output from a series of model runs.

Micro Saint is a proprietary product and therefore not in the public domain. However, MA&D does offer an academic discount for both student and the 'industrial strength' versions of Micro Saint. They also provide excellent training in their product (as well as appropriate courses in the use of IMPRINT).

IMPRINT Development

IMPRINT (Anonymous, 2003 a & b) is a tool developed by MA&D for the Army that helps satisfy part of the MANPRINT requirements Booher (1990). Version 7 uses Micro Saint IV as its underlying computational engine. Version 8, currently under development, will use Micro Saint Sharp as its underlying computational engine, providing some new / enhanced modeling capabilities for IMPRINT that are not treated in this comparison. The Army prefers Law and Kelton (2000) as their basic reference text on simulation and modeling.

IMPRINT has its own graphical user interface and may be used to look at both operator (e.g., individual missions) and maintainer (e.g., sustained combat operation) applications. There are now three levels or modes of modeling that are increasingly complicated and demanding of user input data. All three have 'standard' outputs built-in.

The simplest model implementation permits workload assessments of hierarchical task network models using the McCracken-Aldrich model (1984). The advanced workload assessment mode, restricts the modeling to a single level of task representation, but allows parallel tasking and uses the North model of workload (1989).

The most complex use of IMPRINT uses techniques originally developed under the CART program (e.g., Brett, et al., 2002). This permits goal-directed task modeling that better represents the way in which most missions are accomplished. More recently, the interface of IMPRINT and ACT-R has been explored as well (Kelley and Scribner, 2003).

Both the advanced workload and CART-related modes of IMPRINT give the user access to more powerful modeling tools but fall short of requiring a complete understanding of the full complexities of Micro Saint. This structured support of increasingly more complex models allows new users to systematically develop their modeling expertise.

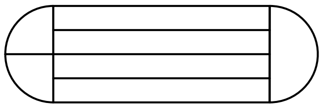
While the documentation does not completely support the user's needs, the Army has made provision to give technical support to new users, something the Air Force did not do for SAINT. This substantially enhances IMPRINT's utility. IMPRINT is a non-proprietary product, supplied free of charge to 'qualified' users – typically organizations with Department of Defense contracts. The point of contact for making a formal request is: Mr. John Lockett, Army Research Labs, Aberdeen, MD.

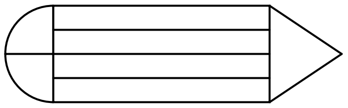
Other Comparisons

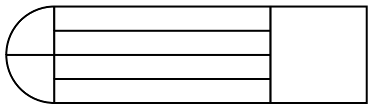
Comparisons can be made on at least four levels: graphic modeling of activities, common features, unique features, and the user interface, for both input and output.

Graphic Modeling

SAINT provided no assistance in developing the network diagrams. A symbol set (Figure 1) was specified for representing task networks, but the diagrams had to be done manually. SADT tools later made this easier to do, but the translation into SAINT was neither direct nor automatic. Micro Saint and IMPRINT both provide facilities for creating the network diagrams.


Deterministic Node: Take ALL Branches Out


Probabilistic Node: Take One and Only One Branch, Based Upon the Relative Frequency Specified


Conditional Node: Take ALL Branches for Which the Stipulated Condition Is Met

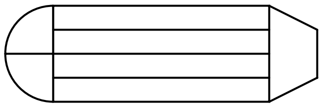

Conditional Node: Take 1st Branch for Which the Stipulated Condition Is Met

Figure 1. *SAINT Symbols*.

The first node in a network starts the task sequencing; subsequent nodes are 'triggered' upon completion of one or more preceding task(s). Directed arrows point from one task node to the task(s) node(s) which follow, and the specified branching logic is applied to determine which path(s) are to be taken. Information packets follow along those paths (like tokens in Petri Nets). These packets are a vehicle for transmission of local information from one task to another (e.g., the level of stress, the value of a control setting, or other definitions for a variable's value). Subsequent tasks can then examine the values of variables passed in a packet. The value can then influence either the time taken by that task, some variable manipulated in the performance of the task, or some condition tested to determine branching out of the task. Micro Saint retained this capability. It provides a very powerful modeling tool.

When any one task completes, one or more of the subsequent tasks may be released for execution, depending on the precedence requirements or release conditions specified for each task. At some point, a terminating node is reached which ends the simulation and initiates the generation of summary statistics for a series of runs / iterations.

When a continuous system's dynamics were modeled, SAINT would also generate a 'strip chart' recording that showed the level (value) of each continuous variable over time, from the start of the simulation to its termination: the time trajectory for each state variable of interest.

The semi-circular left side of all blocks had an upper and lower half specifying what precedence constraints had to be satisfied (what number of preceding tasks had to be first completed before this task was started). In the upper half, one specified how many of the incoming 'signals' had to be present before the current task could be 'released for execution the first time it was performed. The lower half specified how many had to be present before subsequent releases.

Micro Saint did not distinguish between first and subsequent task execution precedence constraints. Task release could instead be specified on the basis of a specified variable, which if 'true' when tested, the task would be released. IMPRINT allows this same representation.

Also, Micro Saint did not use alternate shapes to represent branching alternates. Instead, a dialogue box is presented for the user to select what type of branching is desired. Conditional 'take first' branching is not one of the options however. Prioritized branching can be accomplished through setting and testing variables instead. This scheme simplifies the diagram, leaving details to be specified in terms of data inputs. IMPRINT does the same.

Micro Saint IV and IMPRINT use two different shapes to model functions and tasks. Tasks are always a decomposition (more detailed representation) of functions. Figure 2 shows the older Micro Saint symbology, also used in IMPRINT. Micro Saint Sharp is similar but slightly different. A Node may be either a task or a network of tasks. That network can be either a set of tasks, set of networks of tasks or some combination: a powerful hierarchical approach to modeling complex systems.

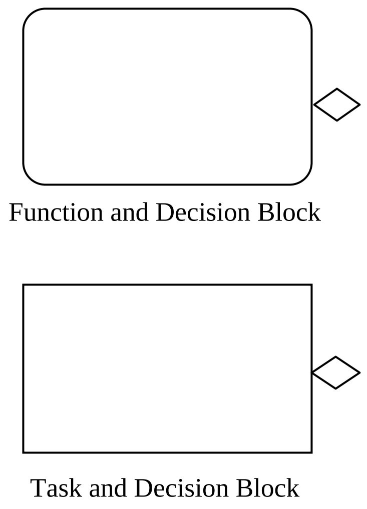


Figure 2. *Micro Saint Symbology.*

By clicking on either the function or the task blocks, the user will bring up the dialogue box that permits entering data associated with the selected function or task. Correspondingly, by clicking on the diamond to the right of each box, the user calls up the dialogue box for specifying the desired branching logic. While this notation simplifies the diagram, it effectively hides the nature of the branching logic.

SAINT did not preclude hierarchical decomposition of task networks, but neither did it facilitate that kind of modeling. Micro Saint distinguishes between upper level functions and the lower level tasks that then support or implement those functions: a network of tasks at one level can appear as a single task at another level. IMPRINT does this too, but in a more limited fashion (a single function layer and a single task layer). However, when using the Advanced IMPRINT workload assessment technique, only a single task layer is allowed, but parallel paths are permitted. The CART-based mode adds yet another consideration: goals drive which functions may be activated at any one time. Several functions may be ongoing at one time, along with their associated tasks. This allows better representation of mission scenarios, but it also is a more complicated form of modeling and typically requires attention to detail and more time in debugging the implementation.

Common Features

SAINT, Micro Saint, and IMPRINT all provide modelers with a wide variety of statistical distributions for representing the duration of tasks or other activities. All three techniques also offer users flexible ways to adjust the parameters of those distributions to reflect the impact of a wide variety of moderators or stressors that change the character of behavior, either in terms of the duration of the task or in the branching that occurs when a task is completed.

Unique Features

System dynamics portray the states of a system (e.g. airplane) continuously over time. The first example used in SAINT was aerial refueling of a B-52 by a KC-135. What was critical was representing the vertical and longitudinal separation between the two airplanes as the pilot changed yoke and throttle settings. Those discrete tasks changed acceleration characteristics, which affected the speed and vertical velocity of the bomber with respect to the tanker, which in turn altered the position of the bomber relative to the tanker (their separation).

While SAINT provided symbols for modeling discrete activities, continuous processes can also be diagrammed, but SAINT assumed users would use either analogue computer techniques (integrator symbols, logic gates, etc.) or the ‘flow rate and level’ symbology used by Forester (1961). For simulation, the differential equations portraying system dynamics are expressed as difference equations for integration of rates to get states. Neither Micro Saint nor IMPRINT provide this capability directly.

IMPRINT on the other hand has a built-in ability to reflect the effects of task accuracy (or, conversely, failure) on performance. This feature appears intuitive on the surface, but users should carefully examine this function to be sure what they think it does is what is actually happening. While the explanations provided seem clear, the user would do well to empirically test a simple model to be sure what they expect will happen actually occurs. Otherwise, they need to reinterpret how this function really works!

The User Interface

SAINT required punched card input, and a single typing mistake meant punching a new card and perhaps rerunning the program. Pre-defined outputs were generated on pre-punched computer paper, not regular 8 ½ x 11 inch sheets. The horizontal format provided more space for printing output, but storage of massive output listings was then awkward.

Micro Saint was designed to operate interactively using a PC’s display screen. Modifications to input could be made more easily, and turn-around for modeling improved greatly. Results could be displayed before printing, so less paper was wasted, and printers started using more conventional sizes of paper. Considerable flexibility is provided to the user in generating output products, including animation.

IMPRINT is more restrictive and directive than Micro Saint and has its own unique user interface, but that also permits more rapid model development with that standardization. The three different IMPRINT modes do have differences in both the input and output interfaces available to users. Each is tailored to the specific mode being exercised, and animation is not provide except in its simplest form: seeing which task(s) get executed. However, this can be quite useful in debugging model implementation.

IMPRINT addresses two important kinds of application: a) system modeling of an individual performing a specific mission, and b) a series of ongoing engagements where the break and fix rates of malfunctioning equipment determine the ability to sustain combat operations. Each use of IMPRINT has its own special characteristics, data input requirements, and output reports.

Conclusions

While SAINT started with the objective of providing engineers with tools that would permit system modeling that also treated human factors, Micro Saint made this approach to simulation and analysis more practical for both engineers and human factors specialists, and IMPRINT tailored the Micro Saint technology to specific needs of the human factors engineer in systems acquisition programs. Micro Saint Sharp has provided a significant advance over the older versions of SAINT but without incorporating its continuous / combined modeling capabilities. While IMPRINT is a bit more restrictive than Micro Saint, it handles workload well, has been extended to treat the degrading effects of a variety of stressors, and addresses some of the impacts training and the lack of practice can have on performance. Version 8 of IMPRINT, now under development, will incorporate some of the features found in Micro Saint Sharp and should therefore be an even more powerful and flexible tool for modeling and analyzing human performance in the context of mission simulation.

References

Anonymous (2003a), *IMPRINT: Improved Performance Research Integration Tool, Analysis Guide, Version 7 DRAFT*, US Army Research Laboratory, Aberdeen, MD, October.
Anonymous (2003b), *IMPRINT: Improved Performance Research Integration Tool, Users Guide, Version 7 DRAFT*, US Army Research Laboratory, Aberdeen, MD, October.

Bloechle, Wendy K. and Daniel Schunk (2003), "Micro Saint Sharp Simulation", *Proceedings of the Winter Simulation Conference*, New Orleans, LA, 7-10 December.

Booher, Harold R., editor (1990), *MANPRINT: An Approach to Systems Integration*, Van Nostrand Reinhold, New York.

Brett, Bryan E., Jeffrey A. Doyal, David A. Malek, Edward A. Martin, David G. Hoagland, and Martin N. Anesgart (2002), *The Combat Automation Requirements Testbed (CART): Task 5 Interim Report – Modeling a Strike Fighter Pilot Conducting a Time Critical Target Mission*, AFRL-HE-WP-TR-2002-0018, Crew Systems Interface Division, Wright-Patterson Air Force Base, OH.

Cellier, F. E., editor (1982), *Progress in Modeling and Simulation*, Academic Press, New York.

Chubb, Gerald P. (1971), *F-106A Nuclear Vulnerability Analysis, Vol. VIII: Crew Effectiveness*, AFSWC-TR-70-7, Air Force Weapons Laboratory, Kirtland AFB, NM.

Chubb, Gerald P. (1989), "SAINT Performance Assessment Model of a SAM System," in Stephen A. Murtaugh and Sally VanNostrand, editors, *Proceedings of MORIMOC II: MORS Symposium on Human Behavior and Performance as Essential Ingredients in Realistic Combat Modeling*, Center for Naval Analyses, Alexandria, VA. 22-24 February, Volume I, pp. 199-219.

Desrochers, Alan A. and Robert Y. Al-Jaar (1995), *Applications of Petri Nets in Manufacturing Systems: Modeling, Control, and Performance Analysis*, IEEE Press, New York.

Forrester , Jay Wright (1961), *Industrial Dynamics*, MIT Press, Cambridge, MA.

Hoyland, Constance M., Debbie Ganote, and Gerald P. Chubb (1988), "C-SAINT: A Simulation Tool Customized for Workload and Information Flow Analysis," Proceedings of the IEEE National Aerospace and Electronics Conference (NAECON). Dayton, OH, 21-25 May, pp. 823-830.

Kelley, Troy D. and David R. Scribner (2003), *Developing a Predictive Model of Dual Task*

Performance, ARL-MR-0056, US Army Research Laboratory, Aberdeen, MD, September.

Law, Averill M. and W. David Kelton (2000), third edition, *Simulation Modeling & Analysis*, McGraw-Hill, New York.

Laughery, “Network modeling on microcomputers,” *Simulation*, January, pp. 10-16.

Marca, David A., Clement L. McGowan (1988), *SADT: Structured Analysis and Design Technique*, McGraw-Hill, New York.

McCracken, J. H. and T. B. Aldrich (1984), *Analyses of Selected LHX Mission Functions: Implications for Operator Workload and System Automation Goals*, Tech Note ASI479-024-84, Army Research Institute, Aviation Research and Development Activity: Ft. Rucker, AL.

Moder, Joseph J., Cecil R. Phillips, and Edward W. Davis (1983), *Project Management with CPM, PERT, and Precedence Diagramming*, Van Nostrand Reinhold, New York.

North, R. A. and V. Riley (1989), “W/INDEX: A Predictive Model of Operator Workload”, In G. R. McMillan, editor, *Applications of Human Performance Models to System Design*, Plenum Press, New York.

Pritsker, A. Alan B. and W. W. Happ (1966), “GERT: Graphical Evaluation and Review Technique – Part I. Fundamentals, *Industrial Engineering*, Vol. XVII, June, pp. 267-274.

Pritsker, A. Alan B. and G. E. Whitehouse (1966), “GERT: Graphical Evaluation and Review Technique – Part II. Probabilistic and Industrial Engineering Applications, *Industrial Engineering*, Vol. XVII, June, pp. 293-301.

Pritsker, A. Alan B. and Nicholas R. Hurst (1973), “GASP IV: A Combined Continuous-discrete FORTRAN-based Simulation Language”, *Simulation*, Vol. 21, September, pp. 65-70.

Prtisker, A. Alan B. (1995), *Introduction to Simulation and SLAM II*, John Wliey & Sons, New York.

Seifert, Deborah J. and Gerald P. Chubb (1978), "SAINT: A Combined Simulation Language for Modeling Large, Complex Systems," *Proceedings of the SIGSIM 78 Conference*, Canberra, Australia. 4-8 September (also identified as AFAMRL-TR-78-48, Wright-Patterson AFB, OH).

Siegel, Arthur I. and J. Jay Wolf (1969), *Man-Machine Simulation Models; Psychosocial and Performance Interaction*, Wiley-Interscience, New York.

Wickens, C. D. (1984) “Processing Resources in Attention”, In R. Parasuraman, J. Beaty, R. Davies, editors, *Varieties of Attention*, Wiley, New York, pp. 63-101.

Wickens, C. D. and Y-Y Yeh (1986), “Multiple Resource Model of Workload Prediction and Assessment” in *IEEE Conference on Systems, Man, and Cybernetics* (Atlanta, GA.)

Yourdan, Edward and Larry L. Constantine (1979), *Structured Design: Fundamentals of a Discipline of Computer Program and Systems Design*, Prentice Hall, Englewood Cliffs, N.J.