VALIDATION -- THE BOTTLENECK IN SYSTEM SIMULATION

Gordon D. Edgecomb
Richard J. Thompson
Chemical Abstracts Service

ABSTRACT

In simulation exercises some degree of validation is necessary in order to create confidence in the results obtained. Experienced users of generalized simulation software and commercially available simulation tools often find that the majority of their simulation efforts are localized in the validation cycle. Discussed in this paper are some of the factors which have caused the validation process to become more complicated and more time consuming. This paper also summarizes the major problems with simulating complex computer systems and suggests some basic requirements and design philosophies for new simulation packages.

INTRODUCTION

The major use of simulation at Chemical Abstracts Service (CAS) is as an aid to computer configuration planning. During the expansion of the computer system over the past five years, CAS's simulation goals have been to predict the impact of changes in workload, hardware, and control system software. The spectrum of project analyses has varied widely in magnitude and complexity. It has included analyzing major model changes in the computer system, forecasting the operational impact of new applications software, identifying system components which have low utilization or negligible growth factors, and answering some very practical computer loading questions.

During the time CAS has been active in simulation work, significant changes have occurred in the company's environment, in the capabilities of the simulation tools which are available commercially, and in the methods employed in the simulation process. Some of these changes have made the simulation analysts' job easier, some have made the job much more difficult.

BACKGROUND

Early simulations at CAS dealt with relatively simple environments. For example, in our first use of computer simulation we evaluated the operation of a fixed batch load on several candidate computer configurations. We used the results of these simulations to aid in the selection of a new computer to replace our IBM 360/65 system. The environment at that time was limited to batch-type processing and the primary metrics under investigation were CPU time, channel load, and memory capacity (the number of jobs that could run concurrently).

The current environment is more complex. CAS is no longer restricted to batch work in an MVT operating system. We now have a mixture of batch, specialized on-line, and TSO applications in a Virtual Storage (VS) operating system. In this environment, the metrics of CPU time, channel load, and memory capacity now become secondary in importance to paging rate, working set size, and response time. Potential changes to the environment include MVS, mass storage, and distributed on-line processing. This will further complicate the simulation task and will introduce newer metrics for consideration such as the utilization of individual resources on an "on-demand" basis as well as staging and destaging time and device considerations.

Changes in computer simulation packages include new attempts to more closely approximate the internal operation of today's computers as well as improved mechanisms for building simulation profiles of the desired workload. The latter change has a significant impact on the batch program model building task by automating this time consuming process (1). However, part of the time that is saved through the use of an automatic model builder is now lost through the increased complexity associated with defining today's computer hardware and software characteristics to the simulator.

Within our range of experience, the basic approach to simulation has not changed. However, the traditional data input and logic building steps are now more appropriately named "Developing Workload Profiles" and "Defining Computer Hardware/Software Characteristics." There are two major reasons for making this distinction. First, the new environment requires a different set of skills for each step and second, the latter step becomes an integral part of the traditional validation loop. The validation loop compares simulated results with empirical data. If the results do not compare favorably, then the simulated computer hardware/software characteristics must be altered and the validation cycle must be repeated. This process continues until the simulated results approximate actual results within acceptable tolerances. Then the model can be considered "calibrated" and subsequent simulations can be made with some level of confidence in the results.

While computer and simulator technology have improved over the years, the simulation validation process has had little attention and support. This "tuning" process continues to be the activity that requires the most effort in terms of human resources. Validation is at the core of many simulation failures and frustrations. It also emphasizes the frailties of the simulation education process and the complexity of today's simulation packages.

SIMULATION INTEGRITY

Armed with formal knowledge and perhaps experience in systems analysis, modeling processes, queuing theory, and the mysteries of "black box" software, the simulation analyst feels he is prepared to answer the real world's question, "What if...?" Instead, the analyst may be faced with a downfall as he is confronted with the practical realities and intricacies of the simulation process. The downfall doesn't come during the drudgery of data gathering or in working out the logical subtleties of the model. It comes when the output is presented and the user, client, or customer asks the next question, "How good is the answer?"

Over time, there are advancements in technology and analysts become more proficient in generating models and developing generalized simulation software. But over the same time, circumstances arise which make it more difficult, if not impossible, to ascertain the reliability of simulation results with any guarantee of accuracy, confidence, or credibility. Consequently, the analyst now finds that more time and effort must be spent in the validation cycle to establish the integrity of the simulation results. Indeed, there are occasions where the inability to complete the validation cycle forces the abandonment of the whole simulation exercise.

The causes of the validation dilemma cannot be attributed to any single factor or chain of events. Our experience indicates that the difficulties contributing to the validation problem exist in three critical areas: simulator complexity, data availability, and personnel skill requirements.

SIMULATOR COMPLEXITY

It's a simple truth that today's simulation software is more complex because the computer systems themselves are more complicated. Commercially available or home-grown software has matured to the point of adequately simulating hardware components in a relatively simple batch system. With this software, the impact of model changes or workload variations can be predicted -- and validated within a reasonable degree of certainty. Today's more sophisticated computer systems not only feature advanced, integrated circuits with faster components, but also involve highly complex control software. A combined environment of batch, time sharing, and other real time applications operating under the control of a VS operating system is not uncommon.

This shift in emphasis from hardware to control software leaves batch-type simulators with a severe handicap. It is not enough for them to treat the operating system as a simplified black box when 50% of the CPU cycles are caused by operating system functions in a VS configuration. This amounts to simulating half a computer model. One half of the model is difficult to validate, and the other half is impossible to validate.

The emphasis on control software also changes the focus on the types of questions asked of a simulation. The concern of DP management now is to determine how to make the best use of this operating system overhead and thus defer the time when more hardware acquisitions need to be made. The objective of the simulations is now optimized throughput, and the questions are in terms of the desired number of active initiators, their priorities, and changes caused by varying input/output (I/O) rates. Internal queuing, paging activity, and their causes and effects are also of major concern. Since the operating system black box approach does not allow access to these specific metrics -- either for manipulation of the model, or for validation of the results -- the simulator algorithm must be made more detailed and more explicit. This of course leads to more complicated simulator software and thus completes the circle of system complexity.

There is an independent development in technology which has not made systems more complex, but has magnified the problem of validating simulation results. This independent development includes the increased and widespread use of hardware monitors, machine accounting, and resource utilization systems. Through the use of this technology, many installation personnel are now more familiar with the operation of their systems than ever before. This increased awareness of computer performance characteristics focuses more attention on the validation process. More metrics are visible and more opportunities exist to validate the specific situations being modeled. Users no longer accept general answers based on loosely-validated general models. They expect the validation process to compare a selected subset of models against the empirical data which is available to them. Unless this comparison is within some acceptable tolerance at this level of detail, the users will place no confidence in the results of a larger-scale simulation.

DATA AVAILABILITY

It's well known that the availability of empirical data is essential to the simulation process. But, the simulation analyst soon finds that the classical sources for this data, namely, system documentation and the analyst who designed the system, are not adequate for his purposes. Systems documentation was never designed to support simulation projects except at a very general level, and thus it rarely contains the detail that is necessary to construct a model. It is often out-of-date and does not coincide with the current operation of the system. Most systems analysts are deeply involved in designing a system under severe time constraints, or they have been reassigned to another project by the time the simulation interview is ready to take place. In either case, the analyst has neither the time nor the recollection that is necessary in order to define an accurate model of the system.

The most reliable and available sources of empirical performance data that can be used for model generation are found in three places:

- Data base access and file usage statistics
- Computer-oriented performance data
- Computer-produced machine accounting data

The data base statistics can provide detailed information on file accessing patterns and data element usage. Hardware and software monitors gather performance data on equipment, and internal software, such as the IBM System Management Facility (SMF) feature of the operating system, supplies the machine and job accounting data. In most cases, all these sources of data must be used for building models of complex computer systems. These sources of data are also vital in the validation cycle.

Organizing this supply of data into terms and structures meaningful to the simulator software has long been a time-consuming manual and intellectual process. Advances have been made in this area, however. Improved mechanisms exist for accepting this data in machine form and automatically building workload profiles for input to the simulation software. But, much of the intellectual effort still remains. Due to either the lack of specificity in the model building algorithms, or to a batch orientation in the simulator itself, a considerable effort in hand-coding models is still required for some applications. For example, if the automatic model builder does not or cannot distribute on-line or TSO transactions over the "connect" time of the terminal, the simulator will receive transaction models that look like batch processes. As a result, the simulator will process a model of typical file records and report that eight hours of on-line transactions can be processed in one minute. Needless to say, the validation process will have little success in correlating these results with actual performance. To eliminate this problem, the model of each type of on-line transaction must be coded separately. For those installations which perform the majority of their work in an on-line mode, the volume of manual model preparation required to handle the message traffic can become almost overwhelming.

PERSONNEL SKILLS

Simulation is not a programmer's tool; it is a specialist's tool. It requires the combined skills of an applications programmer, systems analyst, statistician, systems programmer, and hardware specialist to fully utilize its capabilities. Application of these skills begins with the model building process, continues through the simulation exercise and validation cycles, and ends only when the simulation results are accepted.

In the model building process, analyst skills are required to define the problem, to identify essential metrics and the sources of data, and to structure the model. If simulation software is already available, the model must be expressed in terms acceptable to that software. If model building software is unavailable or inadequate, then some programming may be done to extract, translate, and compare the raw data into suitable input for the simulator.

These same skills are applied in the validation process. Since most empirical data is in the form of machine accounting data, the output from the simulator must be interpreted or converted in order to place the measurements on a common base. Further, this base must be in a report form that is familiar to management or other users of simulation. To obtain this common base, it is often necessary to write special conversion programs that will translate the results of the simulation into report formats which can be compared with system utilization reports that are already in use at the installation.

For many simulation applications, defining the computer hardware and control software characteristics requires the knowledge of skilled systems programmers and/or hardware experts who are generally familiar with the interaction of hardware and software on the computer. It is especially important to have an understanding of the various system interactions that are required in order to execute a hardware I/O operation. Also required is a knowledge of the operation of the major queues in the operating system, the overall mix of instructions on their system, and the potential simultaneous processing capability of the files in each system to be simulated.

Simulation does not come easy, even for someone who possesses all of the above characteristics. The analyst must thoroughly understand the simulator itself and the manner in which it processes model inputs. In general, a lack of knowledge about the internal logic of the simulator is the major obstacle in the validation process. During validation, the analyst modifies simulator inputs to bring the results closer to actual experience. Since the workload profile input is based on empirical data, the majority of the tuning involves modifications to the hardware and software definitions. This tuning can be a time-consuming trial-and-error process since some hardware/software definition changes have little effect while other changes may have gross effects on the results of the simulation. Moreover, tuning one aspect to match empirical data may seriously detune another aspect if the analyst doesn't understand how the simulation program is constructed or is not alert for these possible side effects.

Assume, for the moment, that the validation cycle is successful, and the subset of the model problem tracks reality within acceptable tolerances. The analyst must now introduce additional workload profiles, extrapolate new volumes, and/or change hardware/software definitions to simulate the full scope of the postulated problem. Even with full knowledge of how the simulator handles these inputs, there can be no guarantee that the final simulated results are as accurate as the validated case. It is more probable that the results are less accurate since some simulator functions may not have been exercised or tested during validation. This is particularly true when the object of the simulation is an activity which does not yet exist and for which calibration data is unavailable. And yet, this is exactly the case where a good simulation can be most helpful to a designer or decision-maker.

The problem of educating simulation personnel can be separated into two levels: one, learning to accurately define and code workload profiles and configuration definitions and two, learning to manipulate the simulator to achieve the desired results. While the first level is concerned with the basic techniques required to adequately represent a computer application in simulator terms, the second is concerned with manipulating the logic of the simulator itself in order to insure that it is representing a life-like situation. Training for the first level is inadequate in that formal education does not really address the topic except at a theoretical level. Training in the second level is non-existent in vendor-supplied packages and is a lengthy on-the-job learning process for proprietory software.

SOME POSITIVE STEPS

There are a number of general problems facing the simulation analyst today. Most experienced users anticipate difficulties in system complexity, data gathering, or skilled resources and take steps to reduce the impact of these problems. However, it is interesting to note that they do not always attack the real issues. Contrary to the conventional wisdom, it is not the complexity of the application which causes difficulty; it is the complexity of the simulator software itself. The effort required to gather data for model building is far less than the effort required to validate the results. And finally, a host of skills is required, not just a sound knowledge in simulation techniques.

Our experience has been that the impact of these problems is felt more severely in the validation process than in many other steps of the simulation process. Lengthy, and sometimes, unsuccessful validation cycles have the effect of amplifying the already high cost of running simulations, and of increasing the skepticism of those who receive the simulation results.

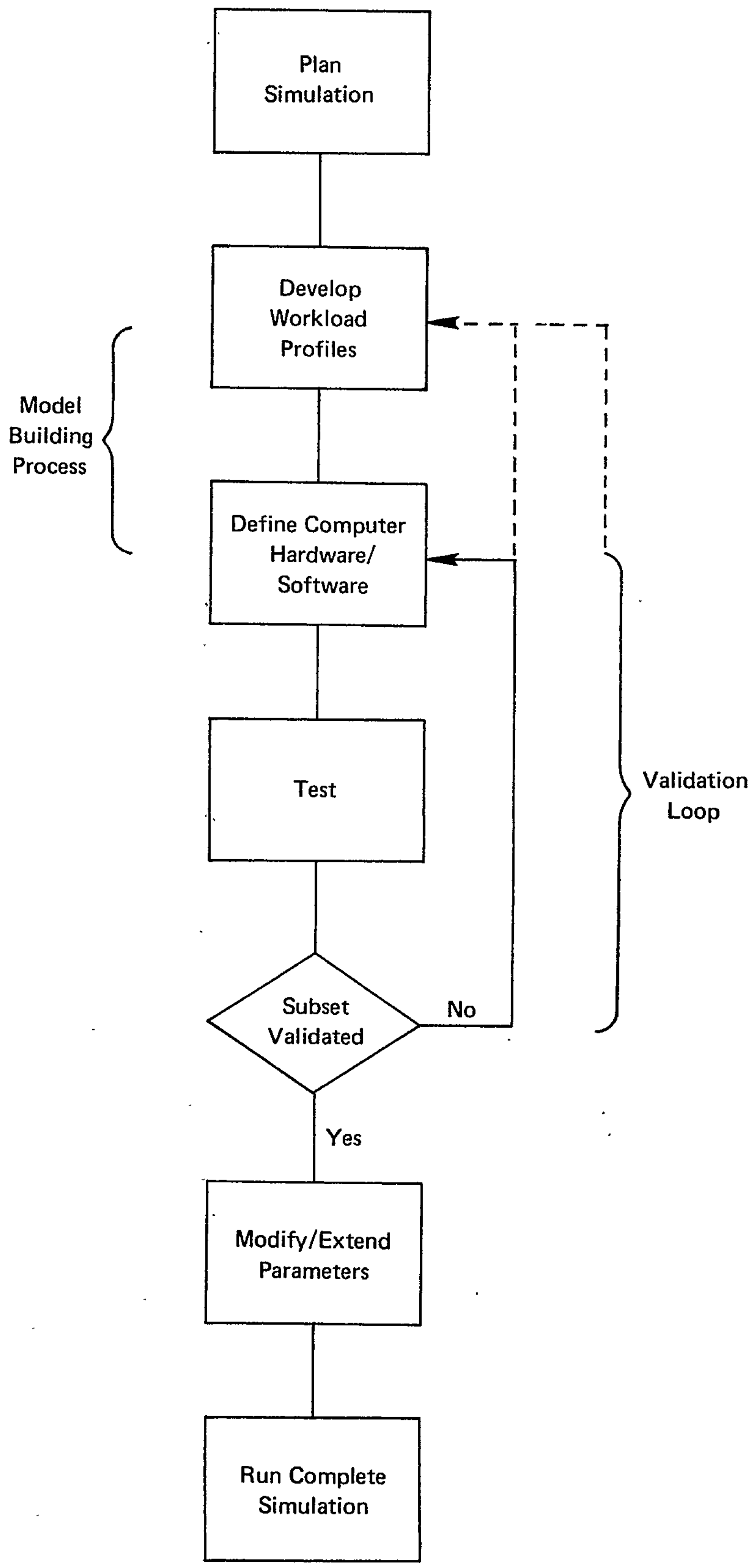
Our experience has also identified some corrective steps which can be taken by the developers of simulator software. The suggestions deal with the general principle of adding more synthetic properties to the simulation software, thus reducing the amount of manual analytical interaction that takes place in today's simulation exercises. Perhaps the ultimate limit in this direction is a simulator which adaptively tracks the solution and reports optimized results.

One step is to develop a set of algorithm-generating programs which could be executed on the simulation user's computer. These programs would act as test drivers to develop service rates of various system queue points under various conditions, thus developing their own queue curves. This approach is similar to that taken on trace-driven modeling techniques as described in earlier papers by Dr. Sherman (2) and by Noetzel and Herring (3). Only the major operating bottlenecks of a particular installation should be considered as candidates for automatic queue curve generation. We also suggest that standardized, machine-readable data sources should be used as input to the algorithm generators. Two sources come to mind -- SMF and hardware monitor data. This approach would eliminate much of the guesswork associated with setting the variable parameters in today's simulators. Of course, these queue curves would be valid only on the particular computer on which the programs were run. The challenge remains for the vendors of simulation software to adapt algorithm generators to the functions of a wide variety of computer models.

The next step is to enable the input of the simulator to be in the same terms as that of the output. Again, we suggest that these terms be in SMF, or some equivalent with which the users are familiar. To assist in the step previously mentioned, simulator pre-processors would synthesize this data into its resource queue demand components for processing by the algorithms of the simulator. This step would immediately reduce the amount of time required in the validation process to resolve differing sets of figures. Further, in the model generating process, hand-coded models would be reduced to an assemblage of SMF definitions.

Automatic model generators are becoming commonplace, but there is still little software available for automatic validation. With the combination of features just described, i.e., SMF type I/O and automatic algorithm generators, considerable progress can be made toward an automatic validation cycle. This automation could occur by comparing the simulated output to original input (both are SMF), modifying the algorithms where required, and executing the simulation again. This iterative process could continue until the simulator was validated according to some parameterized tolerance.

Some attention needs to be given to the typical need to run a sequence of simulations for a sequence of variables in certain input parameters. Simulation is an arduous enough task without having to resubmit runs with only minor alterations to the postulated problem. The ability to accept and


Plan Simulation
Develop Workload Profiles
Model Building Process
Define Computer Hardware/ Software
Test
Validation Loop
Subset Validated
No
Yes
Modify/Extend Parameters
Run Complete Simulation

process ranges of input values on a specified set of variables would add to the flexibility of the simulation process.

We also suggest that designers introduce some type of code to calculate how long a simulation will run based on the input it must process. With all the effort involved in preparation and validation, nothing is more exasperating to the analyst than to see his job abort because it took longer to run than he expected it to.

If these types of improvements were made to simulation software, then the users could devote more of their time to simulation as a tool instead of as an end in itself.

SUMMARY

Although computer and simulation technology have improved over the years, little improvement has been made in the critical simulation validation process. Throughout this paper we have attempted to focus attention on the validation process as an integral part of any simulation exercise and on the factors that influence the result of the validation process. These factors include such items as simulator complexity, contemporary computer system complexity, data availability, and the extensive personnel skills that are required in order to complete the simulation exercise. Along with the identification of problem areas we have also included some suggestions for corrective steps which the developers of simulation software may take in order to reduce the amount of manual analytical interaction required to perform a simulation exercise. Overall, we have tried to present these topics from an experienced user's point of view.

BIBLIOGRAPHY

(1) Edgecomb, G. D., "Automatic Model Building," NTIS, March 1974, PB-232 033/1WC.

(2) Sherman, S., "Trace Driven Modeling: An Update," SIMULETTER, Vol. 7, No. 4, (1976), pp. 87-91.

(3) Noetzel, A. S. and Herring, L. A., "Experience With Trace Driven Modeling," SIMULETTER, Vol. 7, No. 4, (1976), pp. 111-119.