Proceedings of the 2000 Winter Simulation Conference
J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds.

ABSTRACT MODELING FOR ENGINEERING AND ENGAGEMENT LEVEL SIMULATIONS

Robert M. McGraw
Richard A. MacDonald

RAM Laboratories, Inc.
6540 Lusk Boulevard, Suite C200
San Diego, CA 92121, U.S.A.

ABSTRACT

Today's industrial and defense communities are increasingly reliant on the use simulation to reduce cost. At times, due to their stove-piped nature, these simulations themselves have resulted in a waste of both time and money with regard to future simulation development. Current trends address this problem by promoting the development of simulation infrastructures that are scalable, portable, and interoperable over a variety of paradigms. These infrastructures, such as HLA and SPEEDES, address cost issues by providing simulation infrastructures that promote model re-use by managing model interactions across diverse paradigms, improving scenario development, and allowing for a scalable distributed simulation capability.

While these modern simulation infrastructures address many cost-related issues, they do not fully address issues related to model re-use. Simulations that utilize model re-use may result in large complex system models comprised of a diverse set of subsystem component models covering varying amounts of detail and fidelity. Often, a complex simulation that re-uses high fidelity subcomponent models may result in a more detailed system model than the simulation objective requires. Simulating such a system model results in a waste of simulation time with respect to addressing the simulation goals. These simulation costs, however, can be reduced through the use of abstract modeling techniques. These techniques can reduce the subcomponent model complexity by eliminating, grouping, or estimating model parameters or variables at a less detailed level without grossly affecting the simulation results. Key issues in the abstraction process involve identifying the variables or parameters than can be abstracted away for a given simulation objective and applying the proper abstraction technique to replace those parameters. This paper presents approaches for both identifying and replacing these candidate variables.

1 INTRODUCTION

In an effort to reduce developmental and simulation costs while examining complex sets of interactions, present and future simulation development will consider a wide variety of modeling domains and paradigms. For example, the goals for the Joint Modeling and Simulation System (JMASS) program will be to expanded to support other Tri-Service domains for engineering and engagement level modeling. This expansion will result in a simulation system that supports simulation-based acquisition that encompasses a variety of simulation domains utilizing a wide range of commercial tools and application-specific simulations over diverse computational areas. Supported programs may require simulation-based acquisition that supports the design of airframes and power plants, weapons systems and countermeasures. Other programs may require simulation-based acquisition that supports the design of hulls, topside, weapons systems and counter-measures, as well as the design, integration and testing of C4ISR systems (Teknowledge Corporation 2000).

Resultant simulations, such as JMASS, will be comprised of component models of varying degrees of fidelity and resolution that will be used accurately predict system performance, scenario and damage assessments, and mission effectiveness. However, due to these differences in fidelity and resolution amongst models, simulation development may be costly in terms of not only development time but also simulation time (McGraw).

One way to address both the simulation time and development cost issues is to employ model abstraction techniques (Sisti 98). Model abstraction techniques reduce developmental time by allowing re-use of legacy or off-the-shelf models. Likewise, model abstraction techniques reduce simulation time by reducing model complexity.

While being an excellent tool to reduce simulation costs, true model abstraction cannot be achieved by simply "pulling" complexity out of an existing model. Model abstraction techniques must retain information that is key

to determining the performance of a system. Additionally, information abstracted out of a complex model must be properly replaced or characterized in order for the model to remain consistent with the simulation goals. This paper addresses a process for identifying key parameters or variables and replacing or abstracting away that information for models concerning engineering models. Section 2 of this paper discusses some of the simulation objectives that are required engineering and engagement models. Section 3 of this paper discusses methods for identifying key parameters or variables that are critical to the objectives of a simulation. Section 4 of this paper discusses some of the model abstraction techniques that can be used to replace the "non-key" model parameters for models concerning engineering and engagement simulation. Conclusions from this work are presented in Section 5.

2 SIMULATION OBJECTIVES

The objective of abstract modeling is to reduce model complexity without grossly affecting model accuracy with respect to the simulation objective.

2.1 Engineering Level Simulation Objectives

Engineering level simulations are often concerned with system performance. The engineering level simulations can be characterized by the rate at which messages are processed, the amount the system is utilized, or the quickness that the system responds to external stimuli. Specifically, the metrics are characterized as *throughput*, *utilization*, and *response time* (*latency*). The definitions of these metrics are presented by (Lavenberg). Throughput is defined by the equation:

$$X = C/T \quad \text{(Eq. 1)}$$

Where C is the number of completed messages and T is the total time. The *mean service time* per message is defined as:

$$T_s = B/C \quad \text{(Eq. 2)}$$

Where B is defined as the busy time. The utilization is thus defined as:

$$U = XT_s. \quad \text{(Eq. 3)}$$

In other words, the utilization of a component is the product of its throughput rate and the average service time per job. Response time is defined as the time spent in the system. This is also referred to as latency. The response time is computed as the time that the message has completed processing subtracted by the time the processing of that message was initiated.

These performance metrics are delay or latency dependent. In hardware-based systems modeled by engineering level simulation, there are two different types of delay present: explicit delays and implicit delays (MacDonald). Explicit delays are generally associated with the underlying hardware subsystems. These delays include such items as computational delays, processing delays, or message routing delays. Implicit delays often pertain to resource contention issues. These delays may be represented by such items as contention for shared bus structures internal to the architecture or data links. For a given component, variance in these delays may result in multiple potential delay paths that a message may experience while being processed. An example of a hardware component with multiple delay paths is depicted in Figure 1. The delay path that a message may experience may be dependent upon such parameters as message size, message type, component settings, and component state. In assessing the performance of a hardware system, system analysts are often required to characterize these component delays and relate the delays to overall system performance. The determination of these delays, along with how those delays affect throughput, utilization, and response time, are typical simulation objectives for engineering level simulations.

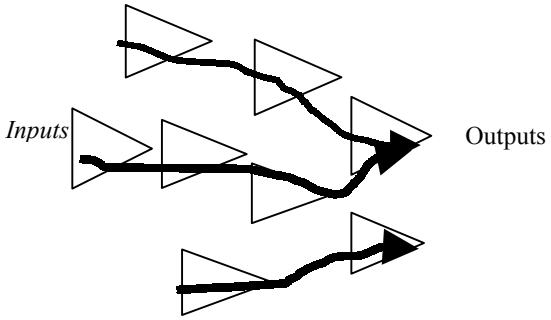

Figure 1: Multiple Delay Paths in Engineering

2.2 Engagement Level Simulation Objectives

Engagement Level simulations represent engagements or encounters between weapons and targets ranging from one-on-one to few-on-few types of scenarios. Typical engagement level scenarios may involve target aircraft with reflecting cross-sections, airborne weapons platforms, RF environments, and airborne missiles with seeker capabilities. Typical metrics derived from these simulation involve determine whether a "hit" or "kill" has occurred for a given set of calibration data. Or, more precisely, developing a "hit" or "kill" distribution over a range of values and calibration data.

3 IDENTIFYING PARAMETERS FOR ABSTRACTION

Model abstraction "captures the essence of the behavior of a model without all the details of how that behavior is implemented in code (Sisti)." Simulations using abstracted models are more concerned with the qualitative results of a simulation rather than the quantitative results of that simulation. Fidelity that is necessary at some levels of modeling may not be necessary to meet modeling and analysis goals at others. For example, the performance metrics associated with engineering level simulation (throughput, utilization and system response) are concerned with message transfers and latencies. Such analysis is not concerned with message content or the actual values associated with various hardware components. For this reason, detail that is not needed to meet the performance analysis goals can be dropped from the model. The process of dropping this unneeded information is known as abstract modeling. Dropping unneeded information allows simulation time to be spent on criteria that is deemed important to the system's operation (Sarjoughian). However, dropping any information may compromise a model's accuracy. "Key" information can be determined using a variety of methods. These methods may range from using a modeler's or analysts' "rules-of-thumb" (or heuristics) or by deterministically determining key parameters using a sensitivity analysis. Some of these methods include extremum experimentation, factorial experimentation, and input sensitization.

3.1 Extremum Experimentation

In the cases where the importance of certain system parameters and inputs are not known ahead of time, *extremum experiments* can be used to identify the key parameters for a given simulation (Dixon). These extremum experiments can be used to identify the system parameters that have the greatest effect on the maximum and minimum performance of the modeled element. Extremum experimentation involves the minimization/maximization of performance metrics (for example, the latencies) of a system component. This optimization process is accomplished by applying specific input vectors to a high fidelity model to identify the input combinations that result in obtaining the longest and shortest delay paths. The problem of selecting these input vectors is often referred to as a *discrete optimization problem* (Parker). Solutions for these discrete optimization problems are not easily obtained. For example, feasible solution spaces are enormous in size, and the solution space grows explosively with the number of discrete choices that need to be resolved. For example, for a model element that requires 200 independent binary inputs, approximately 2^{200} or 10^{60} possible permutations need to be considered. In order to identify key input parameters for such systems, *full factorial experiments* must be used.

3.1.1 Full Factorial Experiments

A full factorial experiment is used to address solutions where all factors (inputs and parameters) must be considered. For such an experiment, $\Pi_n l_i$ experiments are required, where l_i is the number of levels required for factor i. So studying K factors at each of two levels (i.e. if binary-based inputs were examined) requires 2^K experiments, and for T levels, T^k experiments are required. For such a full factorial design, however, as K grows large, the computation expense of performing the full factorial experiment becomes prohibitive. Thus, in order to reduce computational requirements, a *fractional factorial experiment* should be used (Kheir, Walpole, Hogg).

3.1.2 Fractional Factorial Experiments

The design of a fractional factorial experiment may result in requiring only one-half, one-fourth, or even fewer experiments than the full factorial experiment before key input variables or parameters can be identified. The reason that this is a viable experimental strategy is that in many experimental scenarios, certain interactions are negligible. Full factorial experiments that consider these scenarios would waste experimental effort. Using such a fractional factorial experimental process, along with sensitivity analysis can greatly aid the modeler in identifying key model parameters.

3.1.2.1 Sensitivity Analysis

Fractional Factorial experiments can be greatly enhanced through the use of sensitivity analysis. Sensitivity techniques allow the modeler to assess the influence of model input variables or parameters on model output characteristics (i.e. performance) (Iman). These techniques allow the identification of unimportant, or statistically insignificant input parameters. These statistically insignificant inputs are ideal targets for abstraction at less detailed levels of modeling.

Sensitivity analyses can be applied to large complex models displaying the following characteristics: (a) there are many input and output variables; (b) the time to simulate the model is excessive; (c) the model cannot be reduced to a system of equations; (d) discontinuities exist in model behavior; (e) correlations exist among the input variables; and (f) the model outputs are nonlinear, multivariate, time dependent functions of the input variables (Iman). For such models, the model can be defined as a function $Y= f(X_1,....X_k,t)$ of the independent variables $X_1,...,X_k$, and possibly time, t. The variables,

$X_1 ... X_k$, can represent a variety of phenomena within the model. For instance, this may include air speed, air pressure, angle of attack, branch points or different submodels within a larger model. Sensitivity analysis, as defined by (Iman), involves the determination of the change in the response of a model, Y, to changes in individual model parameters, X_i, and specifications. Thus, sensitivity analysis is used to identify the main contributors to the imprecision in Y. However, there does not exist a single algorithm for sensitivity analysis that can be followed from start to finish (Iman). The possible models that need to be considered and the potential problems that can arise are both too diverse to permit such a simplistic approach. The most robust approaches, however, utilize the following two techniques:

1. A Preliminary Variable Assessment
2. A Determination of Relative Variable Importance

3.1.2.2 Preliminary Parameter Assessment

An initial assessment of system parameters is often useful to determine the most influential inputs for a given hardware component. For example, if the resources for measuring inputs are limited, this screening procedure can be used to determine which inputs should receive the greatest portion of those experimental resources. Several commonly used screening techniques are: subjective, differential sensitivity analysis, one-at-a-time design, rank order correlation, and adjoint methods.

The *subjective methods* involve the modelers and investigators working together to discard inputs thought to be unimportant. These methods are sometimes referred to as heuristic methods or rules-of-thumb. This method is the least scientific and is prone to personal biases, although it may be necessary to reduce a large number of input possibilities. The use of this method may necessitate creating an experiment in which one could check for inadequacies of the initial screening decisions (Downing).

Differential sensitivity analysis requires that one calculate the partial derivatives for each input variable. The sensitivity coefficient a_j, is defined as the partial derivative of Y with respect to the input X_j. That is:

$$a_j = \partial Y / \partial X_j \qquad \text{(Eq. 4)}$$

Assuming that Y is linear in X_j, we can estimate the sensitivity coefficient by the ratio of the percentage change in the output Y from its nominal value (the value of the output when all of the inputs are set at the nominal value) to the percentage change in the input X_j from its nominal value and treat a_j as an estimate of the sensitivity coefficient (Downing).

An extension of the differential sensitivity method is to estimate sensitivity coefficients in the *one-at-a-time* design, where each input is evaluated at its mean and then at its mean plus or minus some multiple of its standard deviation (typically $\mu + 4\sigma$). The information from the one-at-a-time design can be used to rank the input variables as to their effect on the output.

The *adjoint* method (Conover) provides a rigorous mathematical method for sensitivity analysis. This method yields the exact sensitivities by determining the sensitivity coefficients for any value of the X_j's. In this case, no assumption is made about the linearity of the input/output relationship.

3.1.2.3 Techniques for Determining Relative Variable Importance

If a technique from the preliminary parameter screen can be utilized to mark a subset of the input parameters as important, it is often desirable to rank these parameters in order of their importance. A number of correlation methods can be used to rank variables. These methods include the Pearson product-moment correlation coefficient method, the Spearman rank correlation coefficient method, the partial correlation coefficient method, Smirnov tests, and the use of standardized regression coefficients.

3.2 Extremum Simulation Experiments for Engineering Level Simulation

Taking into account the various methods for assessing input parameters and determining their relative importance, an algorithm has been developed that allows the system analyst to identify the component variables (parameters) that will allow for the establishment of performance bounds on that particular component. Specifically, this algorithm allows the analyst to identify the variables that result in the minimization or maximization of performance parameters. For engineering level simulation, a key performance metric would be latency. This type of algorithm is known as a partial enumeration algorithm.

3.2.1 Identifying Performance Controlling Inputs for the Partial Enumeration Algorithm

For a hardware component with n inputs, the characteristics of a subset, m, of the n inputs may be known through heuristic means (Dixon 88). Thus, values are generated for the m inputs. Given that a subset of the inputs are known (fixed), the value of a specific unknown input of the hardware component may, or may not, affect the component delay. If that input can not effect the component delay, then the input can be treated as a 'don't care'. The complexity of the interpreted input data generation problem is reduced because of the n unknown hardware inputs, only $(n-m)$ of the inputs need be considered when testing the targeted component for

maximum or minimum performance. At this point, a sensitivity analysis is used to identify the inputs that have the greatest impact on component performance. The result of this sensitivity analysis is the partitioning of the unknown hardware input parameters into two distinct groups: Performance Controlling Inputs (PCIs) and Non-Performance Controlling Inputs (NPCIs). Because the NPCIs do not greatly affect the performance of the hardware component, these variables (parameters) become targets for abstraction.

3.2.2 Sensitivity Analysis of Hardware Inputs

Sensitivity analysis is used to partition the hardware inputs into two disjoint sets: PCIs and NPCIs. The sensitivity analysis presented is in the form of Bernoulli trials. In other words, the PCIs are not ranked in order of importance, they are merely declared as being part of the set of PCIs, or not part of the set of PCIs. The implementation of Bernoulli trials consists of the application of four test vectors to the C4ISR hardware component inputs. Collectively, all steps required to implement a single Bernoulli trial on one input will be referred to as a *single-bit-test*. The input that is being tested will be referred to as the *input-under-test*. There are three steps in the *single-bit-test* procedure (MacDonald).

Step 1: Generate two latency test vectors for the input-under-test. The two latency test vectors are identical, except that the value of the bit pertaining to the input-under-test (a hamming distance of one). In the first test vector, the input under test is assigned a value of '0', while in the second test vector, the input under test is assigned the value '1'. All other unknown hardware inputs are generated from a Bernoulli distribution where the probability of a bit being a '1' is set at 50%. The input values, other than the input-under-test, are identical in both vectors.

Step 2: Apply the set of four test vectors to the hardware component.

(a) Apply reset vector. The reset vector is needed so events on the output lines of the hardware component are visible. These output events must be visible so the latency through the hardware component can be measured.

(b) Apply test vector 1 and record latency. The first latency test vector generated in step 1 is applied to the circuit after the reset vector. The latency for the hardware component is recorded. This latency is referred to as l_1.

(c) Apply reset vector. The reset vector is re-applied to the hardware component.

(d) Apply test vector 2 and record latency. The second latency test vector generated in step 1 is applied to the circuit after the reset vector. The latency for the hardware component is recorded. This latency is referred to as l_2.

Step 3: Compare the two measured latencies for equality. In this step, the values of l_1 and l_2 are compared. If:

(Eq. 5) $(l_2-\Delta t) <= l_1 <=(l_2 + \Delta t)$

then the input is marked as a performance controlling input. The value of Δt is determined by the analyst that is utilizing the hardware component model.

3.2.2 Example of the Single Bit Test

For the purpose of illustrating the *single-bit-test*, a simple example has been developed for the hardware circuit of Figure 2. All gates of this circuit have a 5 ns. delay. Inputs i_1 and i_3 are known inputs (determined by heuristics) while inputs i_2, i_4, and i_5 are unknown inputs. The test vectors (step 2) and the corresponding results are shown in Table 1 for input i_2. Input i_2 is identified as a performance controlling input after one single-bit-test. For illustrative purposes, the circuit is tested exhaustively (2^n experiments). The latencies depicted in Table 2 indicate that all three unknown inputs are performance controlling inputs. Thus, these inputs should not be abstracted away.

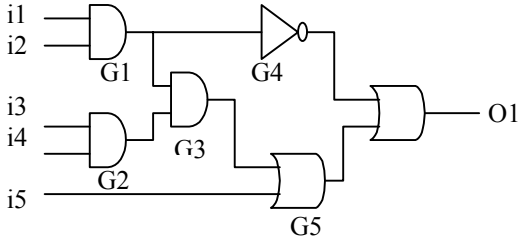

Figure 2: Schematic of Test Circuit

Table 1: Vector Application

Test	i1	i2	i3	i4	i5	o1	Delay
reset	x	x	x	x	x	x	---
vector(1)	1	0	1	1	0	1	15 ns
reset	x	x	x	x	x	x	---
vector(2)	1	1	1	1	0	1	20 ns

Table 2: Exhaustive Test for Unknown Inputs

Known Inputs		Unknown Inputs			Delay
i1	i2	i3	i4	i5	
1	1	0	0	0	15 ns
1	1	0	0	1	10ns
1	1	0	1	0	15 ns
1	1	0	1	1	10 ns
1	1	1	0	0	15 ns
1	1	1	0	1	10 ns
1	1	1	1	0	20 ns
1	1	1	1	1	10 ns

3.2.3 Performance Controlling Input Sets

All performance controlling inputs are classified as either a dependent or an independent performance controlling input. An independent PCI is an input that affects the performance regardless of the values assigned to the other hardware inputs. A dependent PCI is an input that affects the latency of the hardware if and only if there are specific values on a subset of the other unknown inputs. A performance controlling set (PCS) is a set consisting of the input-under-test and a unique subset of the unknown inputs which, when set correctly, will allow the input under test to be detected as such (MacDonald). It is important to note that an unknown hardware input may belong to more than one performance controlling set. The notation for a PCS is to list the input-under-test as the first element of the set. A PCS of size one indicates that the input specified is independent.

The concept of performance controlling input sets can be illustrated using the earlier example of Figure 2. For example, as seen from Table 2, i_5 is an independent performance controlling input. On the other hand, i_2 and i_4 are dependent performance controlling inputs. Input i_2 acts as a performance controlling input when $\{i_4 = 1, i_5 = 0\}$. Similarly i_4 acts as a performance controlling input when $\{i_2 = 1, i_5 = 0\}$. The three performance controlling sets are: $\{i_5\}$, $\{i_2,i_4,i_5\}$, and $\{i_4,i_2,i_5\}$.

In this example, there is only one set of values that can be mapped to inputs i_2 and i_5 that allows the input i_4 to be detected as a PCI. In general, it is possible that there may be more than one set of values that allow the *bit-under-test* to be observed as a PCI. In such cases, it is said that the PCS contains, *a*, active input settings. This is designated as $\{i;x,y\}_a$, where *a* is the number of active settings. When there is only one active setting per set, the subscript is generally omitted. The explicit input values are not shown as part of the PCS. The reason the values are not shown is that both the set size and the number of active input settings per set are important in determining experimental confidence.

3.3 Justification for the Extremum Algorithm

The extremum algorithm that has been presented is exponential in complexity. However, the extremum algorithm is an improvement over full enumeration. For example, a detailed hardware component with *u* unknown inputs would require $2^{(u+1)}$ test vectors if a full factorial experiment is used to identify an extremum measurement. On the other hand, the sensitivity phase of the partial enumeration algorithm requires 2n tests. If the sensitivity analyses detects *k* PCI inputs, a full enumeration of the 2^k patterns is also required. Thus, the partial extremum algorithm requires $2nu+2^k$ tests. This partial enumeration algorithm for extremum modeling requires fewer tests than the full enumeration algorithm as long as *k* is small in relation to the number of unknown inputs.

To demonstrate this point for engineering level simulation involving digital hardware components, a digital benchmark component, ISCAS-85 C3540 is examined. The ISCAS-85 circuits are a series of digital benchmarks used for evaluating algorithms and methods concerning digital hardware. These benchmarks typically fall in the categories of simulation, test pattern generation, and synthesis. Published results for the ISCAS-85 series can be found in (Devadas). Circuit C3540 has 50 primary inputs, 23 primary outputs and is comprised of approximately 1800 gates at the functional hardware level of modeling. Figure 3 depicts part of the C3540 circuit. The primary output O271 is connected to four primary input variables. In other words, the latency metrics obtained from output O271 depend on only 8% of the primary inputs. (A case where the controlling inputs is small relative to the number of overall input variables). Figures 4 and 5 indicate the number of PCIs detected when observing other specific C3540 output parameters. As shown in this particular case for O1450 for Experiment C, after fixing seven of the fifty input variables/parameters, no PCIs were detected. Thus, 43 of the unknown input variables were determined to be statistically insignificant. For model abstraction purposes, these 43 inputs are ideal model abstraction candidates.

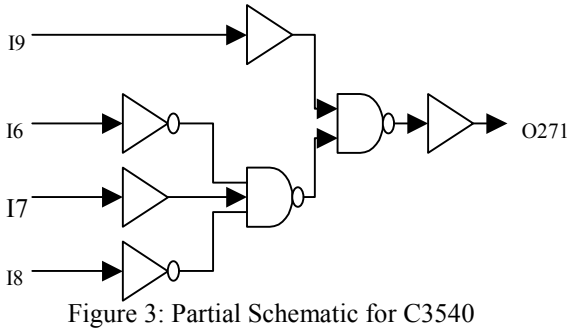

Figure 3: Partial Schematic for C3540

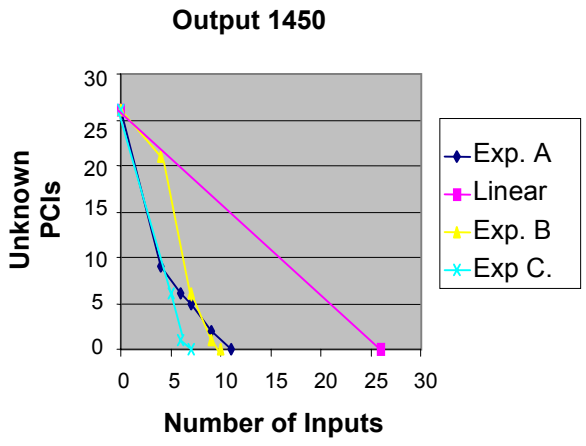

Figure 4: ISCAS-85 Circuit C3540

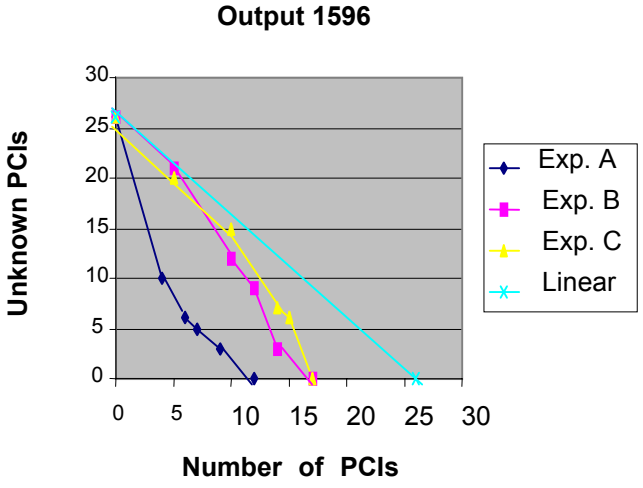

Figure 5: ISCAS-85 Circuit C3540 Output

This process can also be applied to the ISCAS-85 benchmark circuit C2670. Circuit C2670 has 233 primary inputs, 140 primary outputs and is comprised of approximately 1500 gates. Figure 6 indicates the number of PCIs detected when observing C2670 output 1098. When all of the inputs for this circuit are treaded as known, 63 of the primary inputs are found to be performance controlling. This number is only 27% of the total number of inputs. However, exhaustive testing required to identify these inputs would result in 263 input patterns. As shown, as model inputs are identified as critical and become known, the number of performance controlling inputs shrinks rapidly.

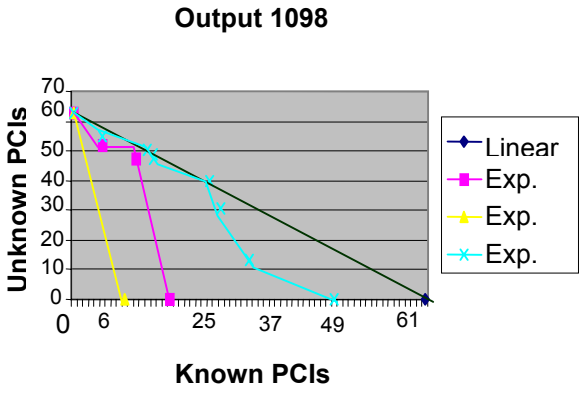

Figure 6: ISCAS-85 Circuit C2670 Output

3.4 Abstract Modeling Techniques

Once input sensitization techniques have been used to identify key input parameters, the trivial input parameters can be abstracted away to reduce model complexity. Several model abstraction techniques may be used for this process. These techniques include the use of stochastic distributions, histograms, data omission, quantizing, aggregation, and other methods. Some of these abstract modeling techniques are briefly described below and are depicted in Figure 7.

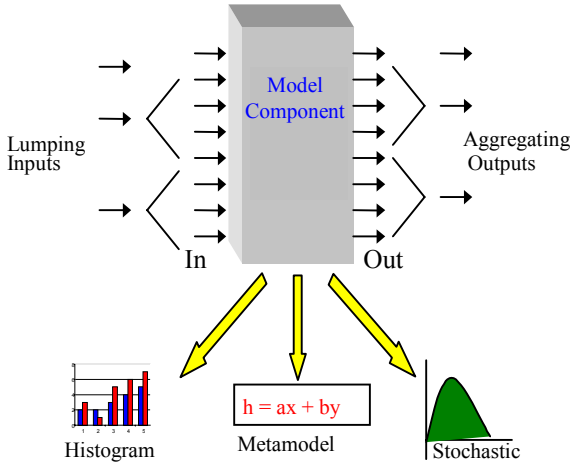

Figure 7: Model Abstraction Methods

3.5 Stochastic Distributions

One technique for developing abstract models of components would be to employ statistical distributions to represent the input/output relationship for the modeled component (MacDonald)(Caughlin). This technique involves developing a distribution for each model output with respect to model input values. Statistical means, variances, and standard deviations can be determined for a

variety of distributions and used to represent the input to output transformation for the modeled component.

3.6 Histograms

Another technique for generating abstract component models is to use histograms (Sisti). In such an example, inputs can be generated and applied to existing models while data is captured at the model outputs. This output data can be used to generate histograms that relate to the prospective input stimuli. These histograms can be used to replace the high fidelity component models.

3.7 Omission

Omission techniques have been presented to abstract information away that is deemed to be "unimportant" (Sisti)(Caughlin)(Zeigler). There are several types of omission methods that can be used to develop abstract models. Omission abstraction methods may involve removing variables from the component model. This process may involve fixing variables to constant values (such as setting "reset" pins for digital engineering models) or simply removing variables from a model, such as removing a degree of freedom variable (i.e. axial spin) from a missile model when considering engagement level simulations.

3.8 Quantizing Parameters

Quantizing inputs involves rounding or truncating internal component or model input variables (Sisti). Quantizing may also involve grouping variable ranges into classes and using aggregation techniques. This procedure removes some of the precision from high fidelity models, but also removes complexity that may or may not effect assessments of system performance.

3.9 Look-Up Tables

A fifth technique for generating abstract component models is to use look-up tables to enter model parameters (Sisti). In such an example, potential model data and parameters can be generated in tabular format and used to replace the behavior of existing models. These tables can be used to simulate the interaction of the existing high fidelity model with other components (players) within the system without requiring the simulation of that component.

3.10 Summary of Abstraction Methods

In order to assess the performance of engineering and engagement level simulations, system analysts often may decide to use a number of these abstraction methods to replace detailed component models. The use of these abstraction methods allows the analyst to omit certain features of the system structure or represent other features of the system structure in a gross way. Model abstraction thus allows the analyst to include only the features that have a primary effect on performance while reducing simulation and development time.

4 CONCLUSIONS

This paper presented an approach to identifying candidate input variables and parameters that can be used to guide model abstraction. Particularly, this paper presented a partial enumeration algorithm for extremum experimentation that can be used in conjunction with performance-based simulation objectives for engineering and engagement level simulations. The algorithm involves techniques that utilize single-bit-tests to identify key variables with respect to performance. This partial enumeration algorithm has been demonstrated using several digital benchmark circuits. This paper also demonstrates that once the "insignificant" component input variables and parameters have been identified using the partial extremum algorithm, those variables can be abstracted away using a variety of techniques ranging from stochastic processes, omission, input quantizing and aggregation, and histograms and lookup tables. The model abstraction technique selected should be the one that has the least effect on model accuracy with respect to the simulation objective. Future work in the area of model abstraction will concentrate more on engagement level simualtions.

REFERENCES

Caughlin, D. and A.F. Sisti. "A Summary of Model Abstraction Techniques," SPIE Conference for Enabling Technologies for Simulation Science, April 1997. pp. 2-13.

Conover, W.J. Practical Non-parametric Statistics, 2nd Edition, John Wiley and Sons, Ed., New York, 1980.

Devadas, S. Keutzer, K., Malik, S., and A. Wang, "Event Suppression: Improving the Efficiency of Timing Simulation for Synchronous Digital Circuits", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, Vol. 13, No. 6, June 1994, pp. 814-822.

Dixon, J.R. "Interactive Respecification: A Computation Model for Hierarchical Mechanical System Design" Proceedings of NSF Engineering Design Research Conference, Amherst, June, 1988, pp. 491-506.

Dixon, J.R., "Computer-Based Models of Design Processes: The Evaluation of Designs for Redesign," Proceedings of NSF Engineering Design Research Conference, Amherst, June, 1989, pp. 491-506.

Downing, D.J. Gardner, R.H., and F.O. Hoffman, "An Examination of Resp0nse-Surface Methodologies for Uncertainty Analysis in Assessment Models," Technometrics, Vol. 27, No. 2 May 1985, pp. 151-163.

Hogg, R.V. and E.A. Tanis, Probability and Statistical Inference, 3rd Edition, Macmillan, 1988.

Iman, R.L., Helton, J.C. and J.E. Campbell, "An Approach to Sensitivity Analysis of Computer Models: Part II-Ranking of Input Variables, Response Surface Validation, Distribution Effect and Technique Synopsis," Journal of Quality Technology, Vol. 13, No. 4, October 1981, pp. 232-240.

Iman, R.L., J.C. Helton, and J.E. Campbell, "An Approach to Sensitivity Analysis of Computer Models: Part I – Introduction, Input Variable Selection and Preliminary Assessment," Journal of Quality Technology, Vol. 13, No. 3, July 1981, pp. 174-183.

Kheir, Naim A., Systems Modeling and Computer Simulation, Marcel Dekker, New York, pp. 179-211, 1988.

Lavenberg, S. S. Computer Performance Modeling Handbook, New York, New York, Academic Press, 1983.

MacDonald, Richard A. Hybrid Modeling of Systems with Interpreted Combinational Elements. Ph.D. Dissertation, University of Virginia. 1995.

McGraw, R. M., MacDonald, R.A., Steinman, J.S., Wallace, J.W. and Buchy, D. "Integration of SPEEDES into the JMASS Architecture," Summer Computer Simulation Conference, July 2000.

Parker, R.G., and R.L. Rardin, Discrete Optimization, Academic Press, San Diego, CA, 1988.

Sarjoughian, H., Ziegler, B., Cellier, F., "Evaluating Model Abstractions: A Quantitative Approach," Proceedings of the SPIE Conference on Enabling Technology for Simulation Science II, SPIE Vol. 3369, pp. 59-70, April, 1998.

Sisti, A., "Enabling Technologies for Simulation Science" White Paper.

Sisti, A., Farr, S. "Model Abstraction Techniques: An Intuitive Overview", Proceedings of SCSC '98.

Teknowledge Corporation, Navy Requirements for JMASS To-Be Technical Reference Architecture, January 2000.

Walpole, R.E. and R.H. Myers, Probability and Statistics for Engineers and Scientists, 3rd Edition, Macmillan, 1985.

Zeigler, Bernard P. "Review of Theory in Model Abstraction" SPIE Conference for Enabling Technologies for Simulation Science, April 1998. pp. 2-13.

AUTHOR BIOGRAPHIES

ROBERT M. MCGRAW is co-founder and Vice-President for Research and Development at RAM Laboratories in Solana Beach, CA. In his current position, Dr. McGraw oversees RAM Laboratories efforts concerning JMASS Risk Reduction and Mixed Resolution Modeling. Dr. McGraw received his BS degree in Physics and Electronics Engineering from the University of Scranton, and his MS and Ph.D. in Electrical Engineering from the University of Virginia.

RICHARD A. MACDONALD is co-founder and President of RAM Laboratories, in Solana Beach, CA. Dr. MacDonald is responsible for overseeing all of RAM Laboratories development programs involving the JSIMS, JMASS, EADTB, and AFRL. Dr. MacDonald received his BS degree in Computer Engineering from University of Michigan, and his MS and Ph.D. degrees in Electrical Engineering from the University of Virginia.