INTEGRATING MODELLING AND DATA ANALYSIS IN TEACHING DISCRETE EVENT SIMULATION

Krysztof Pawlikowski
Wolfgang Kreutzer

Department of Computer Science
University of Canterbury
Christchurch, New Zealand.

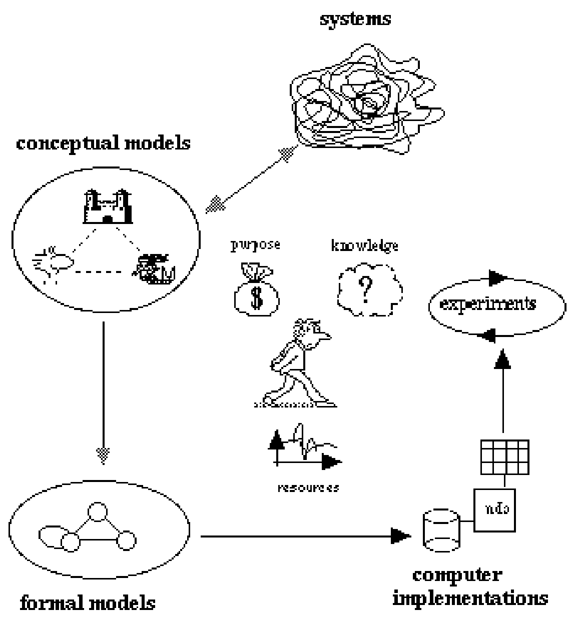
ABSTRACT

The growing popularity of stochastic discrete event simulation in areas such as telecommunication, combined with much marketing hype about ease of use, has coaxed some practitioners into a misguided belief that choosing prefabricated components from libraries and configuring them into a model by pointing and clicking is all that is needed. While neglect of statistical aspects of simulation has already led to some highly problematic published results, this erroneous assumption must also be guarded against in university teaching. This paper therefore argues for the importance of teaching those issues that critically affect the analysis and credibility of a simulation's results alongside those methods and tools targeted at the needs of model design and construction.

1 INTRODUCTION

Using stochastic discrete event simulation successfully requires a valid conceptual model, based on appropriate assumptions. Figure 1 shows the usual steps in a simulation modelling project. The first 4 of these phases (i.e. system identification, model design, model implementation, program verification) have been well researched and documented. Many good textbooks exist, many commercial tools offer convenient programming environments, and the relevant methodologies are often taught well. In order to serve its purpose, however, a model must also be validated and used in a "valid experiment", which requires the application of suitable sources of “randomness” as well as appropriate means of analysing its output data. Both issues are of central importance to a model's credibility and need to be motivated and taught well. Appropriate motivation can, for example, be provided by showing the ease with which inadequate analysis of a simulation's results can lead to erroneous conclusions. Such motivation is particularly important if the relevant techniques for overcoming these problems are conceived as unglamorous and technically difficult. Appropriate choice of convincing examples as well as effective presentation (e.g. by skillful use of visualization and animation of time series data) can be used to overcome this perception.

Figure 1: A Model of Model Construction



Our own work in this context has centered on teaching stochastic simulation of telecommunications networks, but we believe that these issues have wider applicability. Since any stochastic computer simulation must be regarded as a (simulated) statistical experiment, the application of statistical methods of analysis is mandatory.

2 THE GENERATION OF RANDOM BEHAVIOUR

It is a generally accepted practice to use algorithmic generators of pseudo-random uniformly distributed numbers (PRNG) to reflect randomness in stochastic simulation. The theoretical foundations of PRNGs are well established (see, for example Knuth 1998) and over the last 50 years many different PRNGs that pass rigorous theoretical tests have been proposed.

Practically all of these are linear congruential PRNGs (LC-PRNGs) and generate periodic sequences of numbers. The most popular belong to a class of recursive algorithms in integer modulo M arithmetic (Entacher 1998). In today's world of 32-bit computers multiplicative LC-PRNGs with modulus 2^{31}-1 have received special attention and, following exhaustive analysis, about 20 of them can be recommended as acceptable sources for modeling pseudo-randomness (see Fishman and Moore 1986, L'Ecuyer 1990, L'Ecuyer 1991, Park 1988). These are the generators that have been used, for example, in GPSS (version H and PC), SIMSCRIPT II.5, SIMAN and SLAM II (Law and Kelton 1991). As a result one could claim that the search for a good PRNG has become unproblematic.

Unfortunately, this is only partially true. Any conscientious users of PRNGs should be aware that they may face potentially serious problems when using PRNGs in real-life applications. One problem is that recent advances in computing technology have made PRNGs with cycles in the order of 2^{31} effectively obsolete for all but very short simulation runs. Today a standard workstation operating at a speed of a few hundred MHZ can generate the whole cycle of a mod(2^{31}-1) PRNG in a few minutes. And 1 GHz PCs have just been announced (Lewis 2000). When planning a simulation with a runtime of more than a few minutes of CPU time one obviously needs PRNGs of much longer cycles than would have been acceptable only a few years ago. For example, simulations of modern telecommunication networks, fed by traffic streams modeled by strongly auto-correlated processes, need very long runtimes and long streams of output data in order to report results with an acceptably small statistical error.

The use of PRNGs with adequately long cycles is also strongly advocated by recently established theoretical restrictions on the number of pseudo-random numbers from the same PRNG to be used in a single simulation. For example, if one is concerned with two dimensional uniformity of pseudo-random numbers, then, in order to maintain pseudo-randomness of pairs of numbers generated by a PRNG with cycle length L, one should not use more than $8\sqrt{L}$ numbers from a single PRNG during a single simulation (L'Ecuyer 1998, L'Ecuyer 1999b). Fortunately, recent advances have led to the discovery of generators that should be adequate for simulations demanding even very long runtimes for the foreseeable future. For example, a number of Multiple Recursive LC-PRNGs, and Combined Multiple Recursive LC-PRNGs, with cycles between 2^{185} and 2^{377}, have been reported by L'Ecuyer (1999), together with portable implementations. Their pseudo-randomness has been established as satisfactory in up to 32 dimensions.

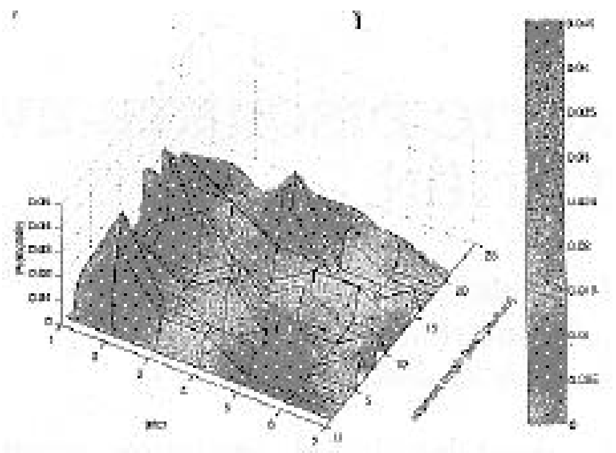
Recently an even more remarkable discovery has been reported. Investigations into a class of Generalized Feedback Shift Register PRNGs (GFSFR-PRNGs) have resulted in the discovery of a *twisted* GFSFR-PRNG, known as *the Mersenne Twister*, with an extremely long cycle of 2^{19937}-1 and good pseudo randomness for 32-bit accuracy in up to 623 dimensions (Matsumoto and Nishimura 1998). Matsumoto and Nishimura's 1998 paper also contains a portable implementation of this generator for 32-bit machines, written in C. This is claimed to be faster than a standard PRNG used in the ANSI C rand() function. See *www.math.keio.ac.jp/matumoto/emt.html* for the latest information regarding the Mersenne Twister.

There are therefore PRNGs of acceptable quality which can serve as practical sources of randomness in stochastic simulations, and their use must be taught. Unfortunately this does not mean that all problems related with PRNGs have been solved. For example, one should be very cautious when using uniformly distributed pseudo-random numbers from a single generator in distributed and/or parallel simulations. The reasons for this lie in potential correlations between disjoint sub-streams of consecutive numbers (Entacher 1998, Hellekalek 1998). As A. Compagner (1995), of the Technical University of Delft (Netherlands) put it : *".. results of stochastic simulation are misleading when correlations hidden in the random numbers and in the simulated system interfere constructively* ..."

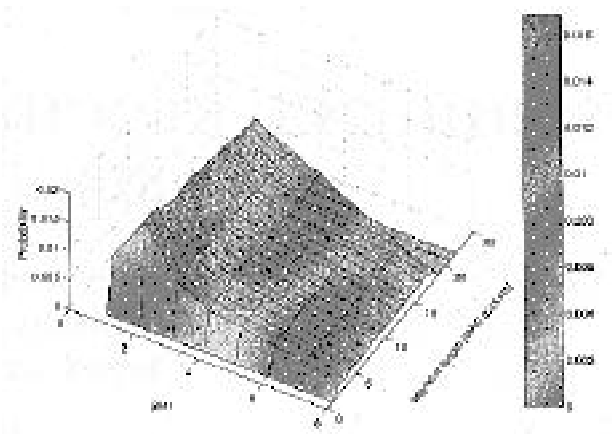
Even in the case of traditional, non-distributed and non-parallel simulation on single processors one must be careful. Uncontrolled distribution of various computer programs has resulted in the uncontrolled proliferation of PRNGs with unsatisfactory or unknown quality. The advice by D. E. Knuth (1998) is even more relevant today: *".. replace the random generators by good ones. Try to avoid being shocked at what you find* ...". Jain (1991) offers a longer list of useful practical guidelines on how to use or not use PRNGs in simulation studies; together with the advice that *".. it is better to use an established generator that has bee ntested thoroughly than to invent a new one*".

3 TEACHING SEQUENTIAL STOCHASTIC SIMULATION

Even where good random generators are used for a model's implementation one must continually guard against any misleading assumption that simulation has now simply become an exercise in computer programming. Successful use of quantitative stochastic simulation for quantitative assessment of dynamic system performance requires more than just the the ability to build useful models. Many respected researchers report that modeling of a simulated system represents only 30-40% of the total effort in most successful simulation projects (Law and Kelton 1991). After a valid simulation model has been designed and a corresponding program has been implemented and verified a researcher still faces the problem of conducting appropriate output analysis. While sadly poorly supported by most commercial tools, this is another skill that novices must be taught. Stochastic simulation should be seen as a (simulated) statistical experiment and analysis of its output data is the necessary condition for any credible final results. If the random nature of such output data is ignored, then ".. *instead of an expensive simulation model, a toss of the coin had better be used*" (Kleinjen 1979). As any other paradigm of scientific research the results of a simulation experiment should be obtained with an appropriately small error. Otherwise they can be misleading or at the least inconclusive.



(a) results with statistical errors of 25% or less



(b) results with statistical errors of 1% or less

Figure 2: Influence of statistical errors on the quality of simulation results.

Figure 2 clearly shows this in case of a model for a Medium Access Protocol of a mobile communication network (Fitzek et al. 2000).

Unfortunately the obvious statistical nature of simulation output data has been neglected to such an extend that one can justifiably talk of a deep credibility crisis in applied stochastic simulation. For example, in the area of telecommunication networks, a recent survey of almost 2300 scientific publications that appeared in a selection of prestigious journals and conference proceedings between 1992-1998 reveals that, while over 50% of all surveyed publications reported results obtained from simulation studies, only about 23% of the simulation-based papers could be considered as credible sources of information which reported statistically analyzed results (Pawlikowski 1999).

One reason, but not an excuse, for this alarming state of affairs may be that the output generated by a typical simulation run can be strongly auto-correlated and the analysis of such time series may require sophisticated statistical techniques. A possible escape from this situation, which can also aid teaching, could employ automated analyses; for which, however, suitable tools must be developed (see, for example Heidelberger and Welch 1983, Pawlikowski 1990).

Statistical errors in simulation results are commonly measured by a confidence interval expected to contain an unknown value. The probability of this to happen is known as the confidence level. In any correctly implemented stochastic simulation the width of this interval will tend to shrink with the number of data points we collect. Two different scenarios exist. The simpler one enters the length of a simulation experiment as an input parameter to the model. Although this method is often defended by arguing that, for “well behaved” models, the output's credibility should improve the longer we run the model, the magnitude of the resulting statistical error is ultimately a matter of luck. While it continues to be a popular “default” it is no longer an acceptable method to teach: *"... no procedure in which the run length is fixed before the simulation begins can be relied upon to produce a confidence interval that covers the the theoretical value with the desired probability*". (Law and Kelton 1991).

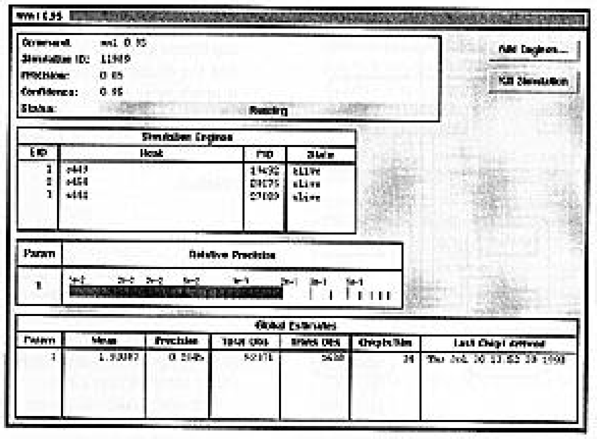
Instead modern methodology offers sequential simulation as an alternative which gives us control over the tradeoff between computational effort and the expected quality of the data we wish to produce. Here a simulation unfolds through a sequence of consecutive checkpoints at which the accuracy of estimates, conveniently measured by the relative statistical error, is assessed. The simulation is stopped at the checkpoint at which the relative error of estimates falls below an acceptable threshold. This method should obviously be the one to be taught. Similar reasoning applies to methods which allow us to approximate a model's behavior in steady state. These require more elaborate statistical methods (Pawlikowski 1990) and suitable motivation and attractive visualizations are therefore of great importance for teaching them well.

The problem with sequential stochastic simulation and any other simulation scenario aimed at obtaining satisfactorily precise results is that modeling even moderately complex models can require very long, or even prohibitively long, simulation runs. To reduce run lengths one could try to reduce the variance of estimators used for the analysis of simulation output. Unfortunately, while many different Variance Reduction Techniques (VRTs) have been proposed (see, for example, Law and Kelton 1991) their robustness and universality have been questioned in practice. An alternative and frequently the only means for shortening run lengths of stochastic simulations is to execute models concurrently, using multi-processor computers or computers linked in a local network. The methodology needed for executing such *parallel simulations* should also be an important ingredient for a simulation curriculum. One possible scenario, known as Multiple Replications in Parallel (MRIP), can easy be applied and has been implemented in the AKAROA-2 modeling tool developed at the University of Canterbury (Ewing et al. 1999).

3. AKAROA-2: AN AUTOMATED SIMULATION TOOL

AKAROA-2 is the latest version of a fully automated simulation tool designed at the University of Canterbury. It is targeted at running distributed stochastic simulations in the MRIP scenario over a local area network. The package has been designed mainly for use with simulation programs written in C or C++, but it can easily be adapted to work with other languages and systems; e.g. a Java port has been built at the University of Hamburg. Akaroa has been used to aid our teaching of simulation methodology for a number of years.

The capability to run existing simulation programs in an MRIP scenario was one of Akaroa-2's main design goals. It accepts an ordinary sequential simulation program and automatically launches the number of simulation engines requested by a given user. Any simulation program which produces a stream of observations and is written in C or C++, or which can be linked with a C++ library, can be converted to run under AKAROA-2. This requires as little as a single procedure call per performance measure to be added to the existing code. Depending on the requested type of stochastic simulation (finite-time horizon or steady-state simulation) appropriate sequences of checkpoints will automatically be generated and a statistically correct method of output data analysis will automatically be applied. The simulation will then be stopped when all results achieve a specified level of relative statistical error; at a given level of confidence . Both of these measures will be specified by the user before the start of a simulation run. To aid its effectiveness in teaching a newer version of AKAROA-2 has been equipped with a graphical user interface (GUI).



Command: mm1 0.95
Simulation ID: 11919
Precision: 0.05
Confidence: 0.95
Status: Running

Simulation Engines

Eng	Host	Pid	State
1	[illegible]	14492	alive
2	[illegible]	[illegible]	alive
3	[illegible]	[illegible]	alive

Param	Relative Precision
1	[illegible]

Global Estimates

Param	Mean	Precision	Total Obs	Trans Obs	Chkpts/Eng	Last Chkpt Arrived
1	1.93087	0.2845	52171	[illegible]	34	[illegible]

Figure 3: How Akaroa-2 shows the status of a model in execution

Figure 3 shows how this interface informs a user about a simulation in progress. The window shows the name of the simulation program (here: mm1 0.95 in its upper left corner), followed by the required level of relative error (or precision) of the results (here: 0.05). The requested confidence level (here: 0.95) and current status of the simulation ("running") is also shown. A table reports the status of the three simulation engines used in this example, followed by a dynamic display of the current relative error and another table displaying the current values of intermediate results. In the upper-right-hand-corner of the window we see two buttons. One is called "Add Engines" and allows a user to accelerate a simulation by increasing the number of participating processors. The other button can be used to stop a simulation before its stopping condition has been reached. More details on AKAROA-2's user interface can be found in Ewing et al (1999).

AKAROA-2 offers fully automated analysis of mean values, both in the case of finite-time horizon and steady-state simulation. The methods of analysis used have been based on an exhaustive survey of their quality, following the methodology presented in Pawlikowski et al. (1998). This research led to adoption of SA.HW.MRIP (a method of Spectral Analysis using the version proposed by Heildelberger and Welch (1981)) and its adjustment to MRIP (Pawlikowski et al. 1994) as the method of automated sequential analysis of steady-state mean values in the MRIP scenario. The length of the initial transient phase is also automatically detected following a sequential implementation of one of the tests proposed by Schruben (1982) for detecting the (non)stationarity of time series.

3 SUMMARY

While there are some notable exceptions (e.g. Simscript II.5, QNAP2, Prophesy ..) sequential stochastic simulation techniques have unfortunately not been at all well supported by vendors of commercial simulation tools, which often rely on the persuasive power of sophisticated graphical presentation and animations instead. While the resulting lack of convenient tools to support teaching these methods poses a challenge, it also offers much opportunity for research.

At the University of Canterbury we have developed a number of teaching tools for this purpose,; e.g. the Akaroa family of simulation engines. These tools have been used for a number of years in both undergraduate and graduate performance modelling courses - particularly targeted at the data communication domain. Due to constraints caused by our institution's degree structures no statistics courses are required prerequisites, but they are strongly recommended as preparation. Our experience with those students who do have a suitable statistical background has been very good. In course evaluations the performance modelling section is considered worthwhile, interesting and challenging. Students without statistical preparation will have to struggle. At the end of the course, however, all participants are well aware of the context in which quantitative stochastic simulations can and should be used, and are able to critically assess the credibility of experimental results. This would not be the case if only model construction and implementation had been taught. While the Akoaroa modelling tools ease application and help to motivate students, they are not essential. What is essential is that a simulation's role as a simulated experiment is stressed and suitable tools for modelling random behaviour and experimental analysis are taught.

In summary we therefore again want to stress the importance of training computer scientists, telecommunication engineers and production planners in how to asses and minimise the errors inevitably associated with conclusions from models which use stochastic simulation techniques. This should be part of any educational program which teaches the use of simulation techniques, regardless of where it is situated or how widely or narrowly it casts its net. Unfortunately existing courses and programs do not always observe this requirement. There is a worrying trend of ignoring critical issues related to statistical credibility, which is also reflected in the relevant literature. A recent survey of publications shows this deficiency clearly (Pawlikowski et al 2000). Let us try to improve this state of affairs by offering our students a solid foundations for **all** phases of simulation modeling and experimentation.

REFERENCES

Compagner, A. 1995. Operational Conditions for Random-Number Generation. *Phys. Review*: 5634-5645.

Entacher, K. 1998. Bad Sequences of Well-Known Linear Congruential Pseudorandom Generators. *ACM Trans. on Modeling and Computer Simulation*: 61-70.

Ewing, G., K. Pawlikowski and D. McNickle 1999. AKAROA.2: Exploiting Network Computing by Distributing Stochastic Simulation'. *Proc. 13th European Simulation Multiconference, ESM'99.* Warsaw: 175-181.

Fitzek, F. H. P., E. Mota, E. Ewers, K. Pawlikowski and A. Wolisz 2000. An Efficient Approach for Speeding Up Simulation of Wireless Networks. *Proc. of the Western Multiconferece. on Computer Simulation, WMSC'2000*. San Diego: in press.

Fishman, G. S., and L. R. Moore III 1986. An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus M = 2^{31}-1. *SIAM J. Sci. Stat. Comput*: 24-45.

Heildelberger, P., and P. D. Welch 1981. A Spectral Method for Confidence Interval Generation and Run Length Control in Simulations. *Communications of the ACM*: 233-245.

Hellekalek, P. 1998. Don't Trust Parallel Monte Carlo! *Proc. of the 12th Workshop on Parallel and Distributed Simulation, PADS'98*. Banff: 82-89.

Jain, R. 1991. *The Art of Computer Systems Performance Analysis*. Wiley.

Kleijnen, J. P. C. 1979. The Role of Statistical Methodology in Simulation. In: *Methodology in Systems Modelling and Simulation*. B.P.Zeigler et al., (eds.). North-Holland

Knuth, D. E. 1998. *The Art of Programming, Volume 2: Seminumerical Algorithms* (3rd edition). Addison-Wesley

Law, A. M. and W. D. Kelton 1991. *Simulation Modelling and Analysis.* McGraw-Hill

Levins, P.H. 2000. With 2 Chips, the Gigahertz Decade Begins. *New York Times,* March 9.

L'Ecuyer, P. 1990. Random Numbers for Simulation. *Communications of the ACM*: 85-97

L'Ecuyer, P. 1998. Uniform Random Number Generators. *Proc. 1998 Winter Simulation Conf.. WSC'98*. Washington: 97-104.

L'Ecuyer, P. 1999. Good Parameters and Implementations for Combined Multiple Recursive Random Number Generators". *Operations Research*: 159-164

L'Ecuyer, P. 1999b. Tables of Linear Congruential Generators of Different Sizes and Good Lattice Structure'. *Mathematics and Computation*: 249-260.

Matsumoto, M. and T. Nishimura. 1998. Mersenne Twister: a 623-Dimensionally Equi-distributed Uniform Pseudo-Random Number Generator*". ACM Trans. on Modeling and Computer Simulation*: 3-30.

Park, S. K., and K. W. Miller. 1988. Random Number Generators: Good Ones are Hard to Find. *Comm. of the ACM*: 1192-1201.

Pawlikowski, K. 1990. Steady-State Simulation of Queueing Processes: a
Survey of Problems and Solutions. *ACM Computing Surveys*:123-170.

Pawlikowski, K., V. Yau and D. McNickle 1994. Distributed Stochastic Discrete-Event Simulation in Parallel Time Streams. *Proc. of the 1994 Winter Simulation Conference, WSC'94*. Orlando: 723-730.

Pawlikowski, K., D. McNickle and G. Ewing 1998. Coverage of Confidence Intervals in Steady-State Simulation. *Journal of Simulation Practice and Theory*: 255-267.

Pawlikowski, K. 1999. Simulation Studies of Telecommunication Networks and Their Credibility. *Proc. 13th European Simulation Multiconference, ESM'99* Warsaw: 349-355.

Pawlikowski, K., Jeong, J. and Lee, R. 2000. On Credibility of Simulation Studies of Telecommunications Networks. Paper submitted to *IEEE Communications* (April 2000)

Schruben, L. W. 1982. Detecting Initialization Bias in Simulation Output'. *Operations Research*: 569-590.

AUTHOR BIOGRAPHIES

KRYSZTOF PAWLIKOWSKI is an Associate Professor (Reader) in Computer Science at the University of Canterbury, Christchurch, New Zealand. He received his PhD in Computer Engineering from the Technical University of Gdansk, Poland. The author of over 90 research papers and four books; his research interests include stochastic simulation, cluster processing, performance modeling of ATM, optical and wireless telecommunication networks, and teletraffic modeling. Professor Pawlikowski is a Senior member of the IEEE. His email and web addresses are krys@cosc.canterbury.ac.nz and www.cosc.canterbury.ac.nz/~krys .

WOLFGANG KREUTZER is an Associate Professor of Computer Science at the University of Canterbury in New Zealand. His current research centers on software design, simulation modeling tools, visual programming languages and computer science education. Professor Kreutzer's email and web addresses are wolfgang@cosc.canterbury.ac.nz and www.cosc.canterbury.ac.nz/~wolfgang.