Proceedings of the 1988 Winter Simulation Conference
M. Abrams, P. Haigh, and J. Comfort (eds.)

Easy-to-use simulation "packages:" What can you really model? (Panel discussion)

Chair
Daniel T. Brunner
Wolverine Software Corporation
7630 Little River Turnpike, Suite 208
Annandale, VA 22003-2653

Panelists
Steven D. Duket
Dean B. Foussianes
Peter L. Haigh
Andrew J. Junga
Paul M. Mellema

OVERVIEW

What is a "Package?"

During the past few years, several discrete event simulation products of a new genre have appeared on the market. They are not interchangeable, but they share a common theme in the way they are promoted that is best summarized by the list of "buzz phrases" below.

"No programming"
"Graphical model building interface"
"Model any system in just a few hours"
"Any manager can use it"
"Anyone on the shop floor can use it"

Commercially available packages in this broadly defined category (that is, software that does or could lay claim to one or more of the buzz phrases) include SIMFACTORY, XCELL+, and WITNESS (and others). The term "package" can also be extended in some cases to custom programs or models that provide access to a particular model or type of model to people who are not knowledgeable about simulation.

How do these packages fit in with widely used existing products such as GPSS/H, GPSS/PC, SLAM II, SIMSCRIPT II.5, and SIMAN? All of the tools in this latter category are simulation language processors. Each processes models that can be said to be "written in" the language of the same name as the package. Should they all be tossed in the dumpster as being too difficult or time-consuming to use?

Here are some of the issues that the panel might address. (The term "packages" is used below to refer to the "new genre" tools described above, and "languages" refers to the language processor category.)

Benefits of the Packages

Benefits of the packages include ease of use, speed of model development, and "sizzle."

Ease of Use. One can hardly dismiss some of the ideas mentioned in the list of buzz phrases. The idea is to make the modeler's job easier. The languages, for instance, enforce cruel and unusual syntax that shows traditional mainframe simulation at its worst, while the packages in some cases have no syntax at all. And real-time display of the process of running the model is a big improvement over watching (minutes or hours, with some PC products) for the cursor to come back.

The packages, as a group, are harbingers of tomorrow's simulation environment. The ease-of-use of a true PC-style user interface should take over the simulation world just as thoroughly as Borland International revolutionized the PC programming environment with the Turbo language family.

Speed of Model Development. When the problem is appropriate to the tool, the "model builder" approach clearly offers significant benefits over a language system. Who wants to type in statements and remember syntax if it is possible to drag a few icons into a network and select "go?"

Sizzle. The world increasingly demands pull-down or pop-up menus, on-line help, fancy graphics, and whatever other sizzle developers can muster. Who hasn't been impressed upon seeing the way the "lines of compiled code" statistic flashes by in Turbo Pascal? Or by how quickly and crisply Brief opens up a new file to be edited? Users (the author included) are frequently inclined to make a 30-second judgment of a new product based on its apparent "pop." If the consumer of the information produced by the model encounters and is impressed by the tool that was used, his or her confidence in the results may be higher than it otherwise might have been.

Limitations of the Packages

Limitations of various members of the packages group include speed of model execution, size of job that can be modeled, and flexibility.

Speed of Execution. This important factor affects not only the simulationist's time spent waiting (and therefore wasted) but also his or her willingness to perform extended, statistically meaningful simulation experiments. Can a point-and-click package generate a model that executes as quickly as the fastest language? If none do, could one be modified or redesigned to do so?

Model Size. Can a package model a medium-sized warehouse, as opposed to a single manufacturing cell? This question has more than one facet. One can ask both "Can the software process a large model?" and "Does the point-and-click or no-programming paradigm support modeling a complex system at all?"

Flexibility. This has been the biggest rap against the packages to date. There seems to be a market for quick tools to do rough cut simulations. Unfortunately, it also seems that many people are going to attempt to apply these tools – given their ease-of-use attractiveness and, in some cases, their low prices – to solve some fairly weighty (in complexity) and highly significant (in dollar terms) problems. Will it work? Why or why not, and in what circumstances?

Are the packages, with their higher-level world view (implying more rigid primitives and paradigms for modeling certain system elements), flexible enough to model moderate-sized to large systems in sufficient detail to be useful?

Deceptive Simplicity. Most real world systems are very complex. A modeling package, however, encourages a "clean" modeling approach, where the most troublesome aspects of our "dirty" world (even such basic concepts as server downtime) might simply be assumed away. How much modeling is, or should be, done this way? Are incorrect "level-of-detail" decisions being made now? What would this imply for the consumers of the information produced by the resulting models, and for the reputation of simulation as an accurate problem-solving tool?

Panel Chair's Biography

DANIEL T. BRUNNER received a B.S. in Electrical Engineering from Purdue University in 1980, and an M.B.A. from The University of Michigan in 1986. He has been with Wolverine Software since 1986. Mr. Brunner is a member of IIE and SCS, and was Publicity Chair for the 1988 Winter Simulation Conference.

POSITION STATEMENT

Steven D. Duket
Pritsker & Associates
P.O. Box 2413
West Lafayette, IN 47906

Simulation design applications closely parallel the engineering design process. Both quick prototyping simulation tools and fully capable simulation "languages" have significant roles in this environment.

Initially, numerous system design concepts are developed. Our goal at this stage is to predict the relative performance of the design concepts, resulting in the selection of the design alternatives deserving more detailed analysis (the "finalists"). Typically, the number of alternatives to be evaluated at this stage is large and the time frame to perform the evaluation is short. For this reason, rapid prototyping simulation "packages" (and languages applied at the aggregate level) can be extremely effective because of their ability to produce understandable results very quickly. Models developed at this stage contain many common assumptions which can simplify the comparison of alternatives without undermining the validity of the analysis.

After the finalists are selected, a detailed analysis of each alternative on a more absolute basis is necessary. This analysis will allow us to achieve a truly functional design and to maximize simulation's cost saving benefits. Since we are looking for the best design which can be created under anticipated operating conditions, issues such as control strategies, scheduling procedures, material handling control logic, etc., must be incorporated in the model. Control procedures for manufacturing operations cannot be standardized and optimized simultaneously. Thus, "packaged" models do not achieve our objectives. Rather, fully capable and flexible simulation languages are required to represent these system specific features to produce the best designs.

Panelist's Biography

STEVEN D. DUKET is Vice President of Pritsker & Associates, Inc. He holds B.S.I.E. and M.S.I.E. degrees from Purdue University. Since joining P&A in 1973, Mr. Duket has specialized in the application of simulation techniques to governmental and industrial materials handling and transportation systems. With the aid of these techniques, he has been involved in the analysis of production throughput and resource utilization in the steel industry, the design of coal port and coal processing systems, the evaluation of tanker, barge, refinery, and pipeline scheduling in the petroleum industry, and the analysis of flexible manufacturing systems.

POSITION STATEMENT

Dean B. Foussianes
Software Services Corporations
1260 Eisenhower Place
Ann Arbor, MI 48108

Much attention is currently given to the trend toward simulation "packages" as opposed to traditional simulation languages. This trend is being driven by 1) the desire to get model development into the hands of the end user, 2) the increasing power/price ratio of personal computers and workstations, and 3) the synergy between animation and simulation.

Engineering literature and commercial advertising would lead one to believe that the traditional languages are a thing of the past. Practical experience has shown that this is not true at the current time. For certain models the interactive packages can provide a reduction in development time. However, in most cases, depending on the skill of the modeler and the particular situation, the modeling packages can provide little if any reduction in development time.

I do feel that the interactive modeling packages are here to stay. As the capability and flexibility of these packages is increased and their limitations removed, they will become increasingly practical. Within the next year we will begin to see packages which remove current computer memory limitations and address the flexibility issue by including "programming" capability within the package. A product with these features and capabilities will likely be well accepted and find many practical applications.

In summary, modeling packages in their current state can be helpful in analyzing certain systems where the desired level of detail and system complexity allow their use. However, for the time being, the traditional modeling languages will remain the preferred alternative for large simulations.

Panelist's Biography

DEAN B. FOUSSIANES is the manager of the Advanced Manufacturing Consulting Group at Software Services Corporations. Based in Ann Arbor, Michigan, the Advanced Manufacturing Consulting Group provides simulation and related consulting services to a variety of Fortune 500 clients and is one of the largest independent simulation groups in the country. Mr. Foussianes is an Industrial Engineer by training and has been associated with the simulation field for the past eight years, with prior experience at Electronic Data Systems and GTE Corporation.

POSITION STATEMENT

Peter L. Haigh
NCR Corporation
SER Building
1700 South Patterson Road
Dayton, OH 45479

What Is A Package?

A programming package is more than a language compiler. Any collection of programs which makes the simulation modeler's job easier could be considered a "package."

The most successful simulation packages, in terms of productivity, are those with a limited domain of problems to simulate. The more specialized a package, the more limited will be its applicability. On the other hand, the more specialized a package is, the more productive it can be for users who deal with problems within its domain.

Should languages be abandoned in favor of packages? The answer is no. There will always be a need for simulation programming languages, just as there will always be a need for assembly languages. Use a package if it can do what you need. Use a language for greater flexibility, higher level of detail, and finer control.

Benefits of the Packages

Ease of Use/Speed of Model Development. A package should be interactive, require minimum learning time, reduce input requirements, improve reliability, provide organization and clarity, and simplify output analysis.

Sizzle. A package should not be chosen based on its marketing interface. One must look past the sizzle and pop to determine if the user interface is helpful or whether it gets in the way.

Limitations of the Packages

Speed of Execution. There is no reason why a package cannot generate code which will execute as fast as code written by a person. Packages which provide user interaction (for model modification) during model execution, however, would most likely employ interpretive techniques. These would bear the burden of slow execution.

Model Size. A well designed package would limit model size only by the available storage. How easily one can implement and manage a large model and how much memory is required for a particular sized model depends on the package.

Flexibility. Flexibility is inversely proportional to specialization. A package for modeling computer systems would not work well for modeling a traffic intersection and vice versa. A general purpose package, however, could model either, but less efficiently in each case than the domain specific package. A domain specific package could be designed to model a large problem rather efficiently.

Deceptive Simplicity. Many problems have been analyzed by over-simplified models. If a package forces over-simplification, it is up to the simulationist to detect this and provide the appropriate amount of detail in the model, abandoning the package if necessary.

Panelist's Biography

PETER L. HAIGH is manager of a computer systems performance analysis group at NCR Corporation. He has directed the development of an in house package for simulating computer systems and networks. He has held positions at Electronic Associates, Inc., XLO Computer Systems, Sweda International, and NCR. His research interests are in the area of interactive simulation modeling tools and the instrumentation and monitoring of computer systems for performance evaluation. He is a member of IEEE, ACM, and SCS.

POSITION STATEMENT

Andrew J. Junga
Applied Systems Modeling, Ltd.
P.O. Box 1239
Anderson, IN 46015-1239

General

A primary concern raised is not so much the packages' ability to simulate a system, but rather the apparent results (and therefore, decisions made) from their use by the novice. A novice refers to someone with little or no simulation experience that views the package as a way to spring-board into the field without paying the usual dues in terms of time spent on the learning curve and in creating models.

Secondly, many of the packages are marketed as full-blown solutions to applying simulation with minimal effort. This approach to modeling will entice individuals to use the package and make decisions "based on simulation" that before were based on spreadsheet results. After all, wouldn't the results of a study based on a simulation be far better than the same facts coming from a lowly spreadsheet? This propagates more individuals and the accompanying problems into the novice class previously discussed.

Finally, the degree of detail the simulationist has access to in the modeling process may be limited. Not knowing the underlying routines executing for a given situation in a package could precipitate exclusion of certain logic that may be a key factor in the operation of a system.

Use by the Novice

A novice approaching the first simulation project will undoubtedly have trouble determining what to include in the model. Similarly, problems will arise as to the determination of when valid and satisfactory results have been attained. The novice may be insulated from these concerns to the point of even eliminating the process of iterations in sensitivity analysis because the first run has been made successfully. An arbitrary decision that the results of this first pass are adequate enough to suppress further, more in-depth evaluation later in the project could result in the system failing because of overlooked detail.

There also the need for a certain level of expertise and common sense in simulation. Many times, a computer programmer has failed as a simulationist after being sent off to learn a simulation language. Learning syntax does not necessarily a good simulationist make. The same general rules apply to using a package.

Effortless Simulation

There are normally two to three levels of analysis in a project. The first "rough-cut, quick and dirty" pass is used to provide a feasibility measure, while the last pass is used for debugging and disaster evaluation. Most passes in between are in support of the first or last. While these packages have great appeal in use as a tool for first pass evaluations, most of this work can be done on a simple spreadsheet.

Easy to use is easy to say. Being able to "model any system in just a few hours" will encourage many people looking to get started doing simulation to purchase a package. A manager responsible for a project would view this as a means to have the "unbiased results of a simulation" help justify a decision. While most managers are not aware of all of the scant details, the desire to be perceived as competent, with a viable project, may overshadow otherwise better judgment, resulting in a "quick and dirty" simulation (just for the sake of having one) that overlooks important factors because of a lack of flexibility.

While point-and-click approaches seem quicker than writing code in a language, many situations in an environment are "same-as/except-for" that can be retrieved as pre-written code from past work or created as a macro.

Level of Detail

Representation of complex situations may be awkward to model due to the packages' canned entity representation and lack of knowledge of the underlying routines executing. The near 1:1 relationship of general purpose language statements and the modeled system make this type of modeling very straightforward.

This relationship also exists in modeling machine controllers, where frequently the logic of ladder diagrams can be coded directly into boolean variables for testing in a modeling language (e.g., a series of electric eyes or sensors on a conveyor). The modeling of the time/space relationship of the designed controller logic as it will apply to hardware can uncover many problems in a system. For example, in the case of a power and free conveyor, the controller logic may work correctly given the carrier can travel at the speed of light. A package will assume the conveyor logic is programmed correctly and may disregard the time involved with shuttling a row of queued pallets.

Conclusions

The inappropriate use of simulation poses a far greater danger than not doing any form of simulation at all. While the same can be said of either languages or packages, the appealing attributes of easy to use, no programming required, point-and-click, etc., make the packages more attractive to those looking for a fast and economical method to produce models, potentially opening the door to problems and giving simulation in general a bad rap.

Simulationists who use primarily languages will continue to try to protect their domain and existence by criticizing the packages. They will also continue, however, to work at a level of detail that seems appropriate to the task. It is in this communication/revision loop and iteration process that much of the detail of the project is stirred up and analyzed (hopefully from an unbiased perspective). Without this interaction, the design engineer may be tempted to sit at his/her desk and evolve a system alone, assisted only by the package and potentially limited experience.

No doubt the trend of the future, even for the time-tested languages, will be to migrate to a "less hostile" environment in terms of ease of use, statistical crutches, animation, etc. While these packages may have their place amongst simulation software, it is our responsibility to educate those unfamiliar with simulation of the dangers and limitations they may impose.

Panelist's Biography

ANDREW J. JUNGA is President of Applied Systems Modeling, Ltd., a simulation consulting company. Prior to co-founding ASM, Mr. Junga spent 14 years with General Motors. He received his B.S.I.E. from General Motors Institute in 1979. Mr. Junga is a certified Professional Engineer and is a member of SCS, IIE, and SME.

POSITION STATEMENT

Paul M. Mellema
Mannesmann Demag Corporation
2660 28th Street, S.E.
Grand Rapids, MI 49508

Background

As a means of illustrating the differences and similarities of using a "package" and using a "language," two models of the same manufacturing material handling system are presented. Both were developed as part of an actual project.

The system in question is one for manufacturing car doors. The supplier of these doors wants to minimize inventory (replacing a warehouse with small buffers), shrink batch size, and automate the circulation of racks among presses, buffers, and assembly lines.

The tools used are XCELL (a "package") and AutoMod (a "language" that is especially suited for modeling certain types of material handling systems). Some differences in approach are evident from the problem statements for the two models.

Problem Statement for XCELL model

Given
Daily production requirements for four part types
Work center routings for each part type
Setup, unit cycle, failure, and repair times for each work center

Find
Least-cost combination of buffer size and changeover policy that satisfies production requirements

Material Handling Assumptions
Zero travel times (infinite capacity)
No carrier contention effects
No path contention effects

Results
Each part type needs a buffer of 30 racks
Lot size = 30 racks
Setup for press run of Xs begins when 30 X racks are empty
Setup for assembly run of Xs begins when 30 X racks are full

Problem Statement for AutoMod model

Given
Speed, acceleration, and number of monorail trolleys
Lift travel time
Track layout, including sizes of control zones
Rules for selecting lane to receive or supply a given rack

Find
Number of racks needed at presses and assembly lines to buffer production against material handling delays

Results
Rack starvation does not occur if local buffers are sized to hold 6 racks

Conclusions

This two-model approach is used frequently at Mannesmann Demag, where simulationists have access to several different tools for simulation modeling. Both the "package" approach and the "language" model play important roles at different stages of the modeling process.

Panelist's Biography

PAUL M. MELLEMA is a simulation analyst at Mannesmann Demag Corporation. In that position, he has been responsible for modeling materials handling, manufacturing, and distribution systems. Prior to joining MDC, he held positions in which he developed software standards and procedures for a large military software project; supervised development of software to teach proof discovery heuristics in symbolic logic; and taught philosophy. He received the Ph.D. in philosophy from M.I.T. in 1973.