17th European Symposium on Computer Aided Process Engineering – ESCAPE17
V. Plesu and P.S. Agachi (Editors)

Assessment of Discrete Event Simulation Software for Enterprise-wide Stochastic Decision Problems

Juan Camilo Zapata, Pradeep Suresh, and Gintaras V. Reklaitis

School of Chemical Engineering, Purdue University, West Lafayette, IN 47907, USA, reklaiti@ecn.purdue.edu

Abstract

The increasing complexity of the enterprise has motivated the utilization of mathematical programming methodologies to deal with many of the decisions faced by companies. However, in many cases the combinatorial complexity of the decision problems is exacerbated by uncertainty in many of the model parameters, requiring the use of scenarios or stochastic programming strategies. Unfortunately, the size of practical problems often makes the use of stochastic programming and stochastic dynamic programming infeasible from a computational perspective. This situation motivated the development of a set of computational strategies known as simulation-optimization as an alternative way to help decision makers in obtaining robust and near-optimal solutions. However, the selection of a discrete event simulation (DES) software to implement such approaches involves a balancing act. The ideal product flexible enough to represent the multiple levels of decision making while allowing the use of any optimization methodology, is not yet available. In this work, we report on the implementation of a multi-stage screening process to evaluate and select the most suitable software for simulation-optimization. A set of criteria was developed and a list of 52 commercial packages was compiled. Nine firms were invited to participate in the final stage of the study, which consisted in analyzing the ability of their software to address a generic problem that included all the criteria. An individual analysis of the seven packages that accepted the invitation was conducted and a quantitative comparison is presented.

Keywords: Discrete event simulation, Simulation-Optimization, Integration.

1. Introduction

Many of the decision processes in the enterprise characterized by uncertainty and suitable for probabilistic modeling are being analyzed and synthesized using DES [1]. DES is a technique that accurately captures the stochastic and dynamic features of the system by using statistical distributions and an events handling mechanism rather than processing algebraic equations. However, the use of simulation as a stand alone tool significantly limits the number of alternative solutions that can be considered. This has motivated the use of simulation-optimization, a set of computational strategies that not only exploit the advantages of DES but also provide a structured way to search for robust/optimal configurations in a constrained space [2]. Therefore, on the implementation side the need for simulators capable of interacting with off the shelf optimization routines or user written search methods is on the rise. An example of this is the increasing number of simulators with some level of built-in optimization routines [2]. In our group simulation-optimization have been used for the past decade, but the limitations of the graphically based simulators to interact with other applications have forced us to use C/C++/Java simulation libraries. This approach tends to generate codes that are not easily scalable, expandable or sufficiently user friendly for industrial application. These considerations motivated us to evaluate the integrability and scalability of graphical multi-purpose simulators from a simulation-optimization perspective.
Today's simulators are highly evolved. They automatically handle all the details of the simulation (generation of random numbers, sampling of stochastic variables, scheduling of events, etc) and provide a complete set of probability distributions (including empirical ones) to model any kind of system. Though the industry has a considerable number of players [3], not all of them target the same market. Production systems, health delivery systems, IT, finance, etc are just some of the areas in which the industry has tailored specific products. Though many of these products are highly efficient in terms of their specific usage, once used outside their scope they become very constraining and even completely impractical to use. Also, those products that claim to be multi-purpose, originated to meet the need of a specific field and therefore tend to have "legacy" constraints that are reflected in their simulation paradigm and internal architecture. The high level of non uniformity in the products is reflected in their specific strengths and weaknesses, which make it difficult to rank their "performance" based on a single measure without losing much useful information. Therefore we chose to develop a set of "flexibility" criteria to evaluate each key aspect we consider necessary to allow the integration of a simulation software with any of the optimization techniques used in the simulation-optimization domain, and the scalability of the combined framework. In addition, the ability of the simulator to use historical data was considered due to its relevance for industrial deployment. In order to make the

study feasible from a time perspective we had to focus on these three aspects. Other performance measures such as robustness, model building complexity, built-in report generation capabilities, speed, etc [4, 5], were not part of the study. It is important to highlight that the criteria were selected having in mind not only the needs of the simulation developers but also of final users who may not have a background in DES. Notice that the designation "simulation developers" does not mean software developers. This distinction is important because any functionality can be appended to the most basic software in the hands of a software developer if its source code is accessible. That said, we drew the line between these two types of developers based on two rules: 1. user written code does not require knowledge of the internal dynamics of the source code (e.g. the use of built-in functions makes the process completely transparent to the user), and 2. the integration between the simulator and the user written optimizer can be done without the need to develop an interface (e.g. COM).

2. Evaluation criteria

The following list includes all the criteria included in the study and a brief description of what each one means and how it was used for evaluation.

- *Hierarchical model building:* The software should be capable of addressing different levels of modeling detail and allowed a seamlessly transition between them. This allows the user to represent the actual structure of the system being modeled in the most realistic yet computationally efficient way..
- *Accessibility to elements:* This criterion reflects the ease of addressing parameters within an element, from another element or from a lower language. This allows modeling of interacting elements that are not physically connected.
- *Model reusability:* This criterion assesses how parts of a model can be made available for future model building and sharing across models and users.
- *Inheritance:* Software designed with inheritance concepts in the model building blocks allow the propagation of changes in the parent block to all the children, relieving the user of tracking down every instance of the parent block and making the corresponding change. This criterion reflects the level of implementation of this concept.
- *Creation and manipulation of user defined variables and arrays at various scope*s: The creation of variables/arrays at various levels, such as , model level, block level, flow item level, etc, that are important for logic and data manipulation are assessed. The ease of manipulation (sorting, row/column addition, etc) of these variables and arrays at runtime is also evaluated.

- *Modularity:* This criterion reflects the extent to which the elements are self contained, in other words, how close the paradigm of data encapsulation is followed in terms of variables, parameters and extensions.
- *Usage of resources by elements:* This criterion checks the ability of the elements in the models to use resources for completion of a task. Specifically, the allocation of multiple units of multiple resource types is evaluated.
- *Extensibility of built in elements:* This criterion measures the conduciveness of the software to user defined extensions of built in elements. Conduciveness in this context refers to the ability to access all the events that occur within an element and to alter the system at the time of occurrence of the event.
- *Facility for designing user defined elements:* This criterion reflects the friendliness of the environment for the creation of user defined elements User defined elements can be created in two possible ways: 1. by grouping multiple elements and giving them an identity, and 2. by coding them completely.
- *Quality of user defined elements:* This criterion assesses the friendliness and transparency of user defined elements.
- *Reusability of user defined elements:* This criterion checks how such elements can be made available for future model building efforts.
- *Interaction with spreadsheets:* The ease of interaction during model building (static) and at run time (dynamic) with spreadsheet tools such as MS Excel for data/information transfer is evaluated by this criterion. The use of spreadsheets within the built-in report generators was excluded.
- *Interaction with databases:* The ease of interaction (static and dynamic) with external databases for data/information transfer is evaluated by this criterion.
- *Link to lower language:* The ability to communicate with programming languages such as C, C++, C#, Java, etc (including calls to libraries (e.g. CPLEX) and other applications (e.g. Matlab)) to extend the capabilities of the simulator to implement complex computations (e.g. optimizers) is evaluated. Not only is the ability to interact with these languages considered important but also the ease of interaction.
- *Dynamic updating of queuing policies:* Flexible systems may require changes in the queuing behavior during the simulation at two different levels: in the queuing policy, and in the attribute used to rank the flow items in the queue. This criterion measures the ease of incorporation of these modifications.
- *Updating model structure at run time:* This criterion checks if the software is capable of updating the model with changes in its structure made by the user or an external application at runtime (addition/deletion of elements). These changes may arise from the need to respond to a specific state of the simulation at some point during the model execution (e.g. if the optimizer determines that the addition of processing units is required).

- *User defined routing:* Dynamic routing, defined as the ability to change the path of a flow item based on the current state of the system, is a common feature in today's systems leading to their flexibility. The capability of the software to incorporate user defined routing allows the user to simulate such systems. This criterion reflects the ability and ease of implementing it.
- *Logic driven pre-emption:* Preemption involves appropriating resources from a task in progress. This behavior is commonly modeled based on the priority of the task. However, in real systems it is usually found that preemption occurs based on some type of logic that involves the state of the system. This criterion checks the ability and ease of implementation of such behavior.
- *Running Multiple Simulations:* Often the user may want to run multiple batches of simulations for the same model for statistical analysis (such as confidence intervals and other performance measures). This criterion checks the capability of the software to run multiple simulations according to user defined stopping criteria that involves data collection and runtime processing.
- *Start from non empty state:* This checks the ability of the software to start in a desired initial state.
- *Adaptability to model changes:* This criterion assesses the ease in altering the model structure and parameters that define the behavior of each element during model updating to match changes in the real system.
- *Animation layout development:* This criterion evaluates the ease of animation creation and its visual appeal.
- *Quality of built-in elements:* The variety of functionalities offered by the built in elements is evaluated by this criterion. It reflects the ability to model a system without the need to use extensions.

3. Evaluation of the third phase simulators through the case study

For this stage a stochastic decision problem was designed with two objectives in mind: to serve as a testing ground for each of the evaluation criterion, and provide a realistic context that could reveal limitations of the software not apparent when each criterion is tested in isolation. The problem, which is a generalization of the new product portfolio case study described in [1], was not formulated to a point at which it was operational, but a strategy was developed to implement it in each software and each of the concepts in the strategy were tested. Though a complete analysis of the strengths and weaknesses was performed for each package, due to space limitations these details have been omitted in this paper[1]. The following table summarizes our observations. An absolute scale (no relative comparison) of 0-5 was used, where 0 represents absence of the criteria and 5 represents complete satisfaction of the criteria.

Packages / Criteria	eM Plant v7.6	Extend v7.0	Flexsim v3.5	Micro Saint v2.2	Quest v5 R17	Sim Cad v7.2	Workbench v5.2
Accessibility to elements	5	4	5	3	4	4	4
Creation and manipulation of user	5	5	4	3	3	2	3
Hierarchical model building	5	4	5	5	4	4	3
Inheritance	5	1	0	0	3	0	2
Model reusability	5	5	5	3	5	5	5
Modularity	5	3	5	2	2	2	5
Usage of resources by elements	4	4	1	**	5	4	3
Extensibility of built in elements	5	5	5	**	5	5	5
Facility for designing u.d elements	5	4/2*	5	2	0	0	2
Quality of u.d elements	5	5	5	1	0	0	1
Reusability of u.d elements	5	5	5	0	0	0	5
Interaction with databases	5	5	5	3	0	2	0
Interaction with spreadsheets	5	5	5	3	0	2	0
Link to lower language	4	4	5	5	4	4	5
Updating of queuing policies	5	5	5	0	5	1	3
Updating model structure	5	5	2	0	5	1	1
User defined routing	5	5	5	4	5	4	5
Logic driven preemption	3	4	3	0	1	3	4
Running multiple Simulations	4	5	4	3	5	4	5
Start from a non empty state	2	5	2	2	3	5	3
Adaptability to model changes	5	4	3	2	3	3	3
Quality of built-in elements	5	4	4	**	4	4	3
Animation layout development	5	5	5	2	5	5	2

* H-block based/Code based. As discussed in the analysis Extend supports these two strategies to create user defined elements,** Criteria not relevant to the software

4. Conclusions

This study screened a large number of commercial discrete event simulators based on the particular features required for the implementation of simulation-optimization techniques. As evident from the table, the first three packages clearly emerged as the candidates that addressed the largest number of criteria

References

1. Varma, V. A., Blau, G. E., Pekny, J. F., and Reklaitis, G. V. (2007) "Enterprise-wide Modeling & Optimization - An overview of emerging research challenges and opportunities."Computers&Chem Engrg, 31, pp 692-711.
2. Fu, M. C. (2002). "Optimization for simulation: theory vs. practice." INFORMS Journal on Computing, 14(3), 192-215.
3. Swain, J. J. (2005). "Seventh Biennial Survey of Discrete-Event Simulation Software." ORMS Today, 30(4).
4. Hlupic, V., Irani, Z., and Paul, R. J. (1999). "Evaluation framework for simulation software." Int. J. of Adv. Manufacturing Technology, 15(5), 366-382
5. Tewoldeberhan, T. W., Verbraeck, A., Valentin, E., and Bardonnet, G. "An evaluation and selection methodology for discrete-event simulation software." Proceedings of the 2002 Winter Simulation Conference, San Diego, California.

[1] For a full version of this paper, please refer to http://web.ics.purdue.edu/~zapata