Selecting Simulation Software

Introduction

The purpose of this document is to provide suggestions and guidelines for selecting a simulation software tool that will meet your needs.

Note that the term simulation is used in different ways by different people. As used here, simulation is defined as the process of creating a *model* (i.e., an abstract representation or facsimile) of an existing or proposed *system* (e.g., a project, a business, a mine, a watershed, a forest, the organs in your body) in order to identify and understand those factors which control the system and/or to predict (forecast) the future behavior of the system. It does not refer to live, interactive simulations (e.g., flight or battle field simulators).

The document does not provide a list of features you should look for, since the requirements of each user are likely to be quite different. Rather, it focuses on the overall ***process*** that you should go through in order to select the appropriate simulation tool. As such, it is intended to be applicable to a wide variety of simulation applications.

Overview of the Steps Involved in Selecting Simulation Software

The steps for selecting simulation software are outlined below (and detailed in subsequent sections):

1. Establish the commitment to invest in simulation software to solve your problem.
2. Clearly state the problem (or class of problems) that you would like to solve.
3. Determine the general type of simulation tool required to solve the problem.
4. Carry out an initial survey of potential solutions.
5. Develop a list of functional requirements.
6. Select the subset of tools that appear to best meet the functional requirements.
7. Carry out a detailed evaluation of the screened tools and select a solution.

Step 1: Establish the Commitment to Invest in Simulation Software

Before spending any effort to research simulation tools, the organization should establish the commitment to invest both the necessary money and staff time into purchasing and learning how to use a simulation software program. Depending on the type of simulation tool selected, the price for a single license is likely to be no less than $2000, and could be as much as ten times higher than that.

Note that it is important for the organization to understand that the cost of a simulation tool is not just the cost of the software itself, but the cost to become a fluent user of the software (since staff time has an inherent cost). In fact, given the complexity of the more powerful simulation tools, the investment in time is likely to be greater than the investment in the software itself.

If the issue you are trying to address represents a one-time need, it may be more cost-efficient to hire a consultant to do the work (so that the organization does not need to purchase and learn the software at all). However, if the issue is recurring or ongoing such that the model will need frequent refinement, or if for some other reason it is important to the organization that the work be done internally, it will be necessary to purchase simulation software and train individuals in its use.

Step 2: Clearly State the Problem You Wish to Address

Perhaps the most important step in selecting simulation software is to clearly state the problem (or class of problems) that you would like to address. This must include a general statement of what you would like the simulation tool to do.

Without doing so, it will be impossible to determine, first, the type of simulation tool you should look for, and subsequently, to list the functional requirements and desired attributes of the tool.

To illustrate what is required, several examples of simulation problem statements are listed below:

- *Managing the water supply for a city*. Managing a water supply is difficult due to the dynamic (and naturally unpredictable) nature of the problem (resulting from uncertainties in both weather and demand). The simulation tool must be able to predict the movement of water through a system (e.g., reservoirs, distribution systems) tracking the quantities and flow rates at various locations. It must be able to quantitatively represent the inherent uncertainty in the system (due to the uncertainty in the weather and demand), and represent various management options (e.g., rules for allocating flows under specified conditions). The output of the simulation will consist of probabilistic predictions of daily water levels and flow rates over time given a specified management alternative.

- *Carrying out a risk analysis for a complex mission (i.e., a machine and/or persons performing a specified task or set or tasks)*. Carrying out a risk analysis for a complex mission is difficult due to the complex interactions and dependencies of the various components, and the fact that the environment may evolve dynamically during the mission. The simulation tool must be able to simulate the operation of the machine throughout the mission, explicitly modeling component interactions, dependencies and failures. It must also be able to represent the impact of a changing environment on the components. The output of the simulation will consist of probabilities of failure (and success) for a mission of specified length, and identification of key failure mechanisms.

- *Modeling the financial outcome of several alternative projects*. When selecting or ranking various alternative projects or undertakings, it is necessary to quantitatively evaluate both the costs and revenues associated with each project. The simulation tool must be able to simulate the future costs and revenues associated with alternative projects, explicitly accounting for the uncertainty in costs, durations and revenues. The simulation must be able to represent disruptive events (e.g. strikes, price changes) and resulting contingency plans that allow a simulated project to respond to new developments in a realistic way. The output of the simulation will consist of probabilistic predictions of the NPV and IRR for each alternative.

Note that these statements are not extremely detailed, but provide a clear statement of the problem, a general statement of what processes and features must be included, and what the output of the simulation will be. This provides enough information to direct a survey of potential solutions and carry out an initial screening. In a later step in the process, more detailed requirements will need to be defined in order to differentiate between the available options.

Step 3: Determine the General Type of Simulation Tool Required

Because simulation is such a powerful tool to assist in understanding complex systems and to support decision-making, a wide variety of approaches and tools exist. Before trying to survey all available tools, you must first decide upon the general type of tool that you require.

There are a variety of simulation frameworks, each tailored for a specific type of problem. What they all have in common, however, is that they allow the user to model how a system might evolve or change over time. Such frameworks can be thought of as high-level programming languages that allow the user to simulate many different kinds of systems in a flexible way.

Perhaps the simplest and most broadly used general purpose simulator is the spreadsheet. Although spreadsheets are inherently limited in many ways by their structure (e.g., representing complex dynamic processes is difficult, they cannot display the model structure graphically, and they require special add-ins to represent uncertainty), because of the ubiquity of spreadsheets, they are very widely used for simple simulation projects (particularly in the business world).

Other general purpose tools exist that are better able to represent complex dynamics, as well as provide a graphical mechanism for viewing the model structure (e.g., an influence diagram or flow chart of some type). Although these tools are generally harder to learn to use than spreadsheets (and are typically more expensive), these advantages allow them to realistically simulate larger and more complex systems.

The general purpose tools can be broadly categorized as follows:

Discrete Event Simulators: These tools rely on a transaction-flow approach to modeling systems. Models consist of entities (units of traffic), resources (elements that service entities), and control elements (elements that determine the states of the entities and resources). Discrete event simulators are generally designed for simulating processes such as call centers, factory operations, and shipping facilities in which the material or information that is being simulated can be described as moving in discrete steps or packets. They are not meant to model the movement of continuous material (e.g., water) or represent continuous systems that are represented by differential equations.

Agent-Based Simulators: This is a special class of discrete event simulator in which the mobile entities are known as agents. Whereas in a traditional discrete event model the entities only have attributes (properties that may control how they interact with various resources or control elements), agents have both attributes and methods (e.g., rules for interacting with other agents). An agent-based model could, for example, simulate the behavior of a population of animals that are moving around and interacting with each other.

Continuous Simulators: This class of tools solves differential equations that describe the evolution of a system using continuous equations. Although these tools usually have some mechanism to represent discrete events, they are most appropriate if the material or information that is being simulated can be described as evolving or moving smoothly and continuously, rather than in infrequent discrete steps or packets. For example, simulation of the movement of water through a series of reservoirs and pipes can most appropriately be represented using a continuous simulator. Continuous simulators can also be used to simulate systems consisting of discrete entities if the number of entities is so large that the movement can be treated as a flow.

Hybrid Simulators: These tools combine the features of continuous simulators and discrete simulators. That is, they solve differential equations, but can superimpose discrete events on the continuously varying system. This can be useful, for example, in business simulations, in which information and material can be modeled as moving continuously, but discrete financial transactions also need to be represented.

Before starting your search for a simulation tool, you should first determine which of these types of tools is required to solve your problem. In most cases, this can be determined from the problem statement. If you are unsure, you should seek input from someone who is familiar with simulation modeling (e.g., a consultant). One of the worst mistakes you can make is to select the wrong type of tool (e.g., to select a continuous simulator, when what you really need is a discrete event simulator).

Step 4: Carry Out an Initial Survey of Potential Solutions

Once you have selected the general type of tool you will need, you can then carry out an initial survey to try to identify the possible options. Note that this process does not involve actively evaluating any software tools. It is simply a survey to see what options are available. The only screening that should be carried out should be based on general type. For example, if you have determined that a continuous simulation tool is required, you should screen out pure discrete event simulators.

This initial list of candidate tools can be generated from a variety of sources, including web searches, peer recommendations, advertisements in trade magazines, and vendor lists from trade-shows.

The output of this step is a list of candidate simulation software solutions that potentially meet your needs.

Step 5: Develop a List of Functional Requirements

Step 5 involves developing a set of functional requirements that you would like the software tool to have. This list will then be used in a subsequent step to compare and contrast the candidate solutions and filter out all but the most promising candidates.

A functional requirement is a necessary feature or attribute of the simulation software solution. Note that requirements specify *what* the simulation software will do, not *how*. They should be as concise as possible. You should also note whether a requirement is mandatory or simply desired (e.g., "must have" in a requirement could indicate mandatory; "should have" could indicate desired, but not mandatory).

In order to develop a list of requirements, you generally start with your problem statement, and describe the minimum set of functionality that will be necessary in order for the software to solve your problem. The actual users of the software will be the primary developers of the requirements list, but other stakeholders should also be involved, such as the ultimate client for the model (e.g., a manager) and IT personnel, as they may have their own requirements.

To illustrate what is meant by a functional requirement, let's consider the first example problem statement listed in the description of Step 2 above:

- *Managing the water supply for a city*. Managing a water supply is difficult due to the dynamic (and naturally unpredictable) nature of the problem (resulting from uncertainties in both weather and demand). The simulation tool must be able to predict the movement of water through a system (e.g., reservoirs, distribution systems) tracking the quantities and flow rates at various locations. It must be able to quantitatively represent the inherent uncertainty in the system (due to the uncertainty in the weather and demand), and represent various management options (e.g., rules for allocating flows under specified conditions). The output of the simulation will consist of probabilistic predictions of daily water levels and flow rates over time given a specified management alternative.

The list of functional requirements for this problem statement would likely include the following mandatory requirements:

- Must be able to track and conserve the continuous movement of material through a system (in this case water).

- Must be able to represent random discrete changes to the system (e.g., pump failures)
- Must be able to represent stochastic processes (e.g., rainfall).
- Must be able to represent rules for allocating and splitting flows.
- Must be able to enter time series inputs.
- Must be able to import time series inputs and other data from spreadsheets.
- Must support Monte Carlo simulation.
- Must have a user interface that supports creation of transparent, well-documented models.

Desired (but perhaps not mandatory) requirements might include:

- Should be able to easily handle unit conversions
- Should be able to support distributed processing (for Monte Carlo simulation).
- Should support optimization.
- Should provide tools for sensitivity analysis.

Functional requirements requested by other stakeholders in the decision might include the following:

- Must have a free reader or player that allows third parties to view and run models without purchasing or learning to use the software (a manager's requirement).
- Should support a floating (network) licensing option to make license management easier (an IT requirement).

Although this list is not necessarily complete (a real requirements list for this problem would likely be a bit longer), it should give you an idea of what the requirements list should look like.

Finally, when creating a list of functional requirements, there are several common mistakes that should be avoided:

- Not involving all stakeholders (e.g., managers, IT staff)
- Classifying all requirements as mandatory or high priority.
- Creating vague requirements.
- Specifying requirements that are not really necessary (and no one will use).

Step 6: Select the Subset of Tools that Appear to Best Meet the Functional Requirements

Once you have defined your functional requirements, the next step is to apply the requirements, to the candidate solutions, identifying and eliminating candidates that do not meet the mandatory requirements.

Note that this step should not require downloading and running the candidate software. Instead, the reviewer should be able to gather sufficient information to develop informed yes/no answers to the requirements based on the vendor’s web pages, quick tours, animated demos, white papers, case studies, recorded webinars, and in some cases, phone calls with technical sales representatives. If you cannot easily gather information about a software product, it is recommended that you eliminate that product from consideration (as this is a generally an indication that the quality of the product and/or the level of support is likely to be poor).

The output of this step is a list of viable solutions, each one of which will then be evaluated in greater detail in the next step.

Step 7: Carry out a Detailed Evaluation of the Screened Tools

The final step in the process involves carrying out a detailed evaluation of the tools screened in Step 6 and selecting the most appropriate tool.

To do so, you should obtain an evaluation version of each product and experiment with the software yourself. Although this is necessary, it can be time-consuming, since each product will have a learning curve.

An excellent way to save time in this process is to make the vendors do the work for you. In particular, put together a simplified "test case" and send it to each vendor. Ask them to demonstrate how their software would be applied to the problem. This allows you to see how an expert user would apply the software to your problem. Of course, the test case needs to be small enough so that it is reasonable to ask the vendor to build the model (you can't expect them to spend a week building a model for you; but they should be willing to spend perhaps half a day). If the vendor is unwilling to do this, it is recommended that you eliminate the product from consideration, as this is a generally an indication that the quality level of support is likely to be poor (if they won't provide support to close a sale, it is unlikely that they would so after the sale).

Note that in order to compare and contrast the candidates, you may want to list all of the functional requirements, and rank the degree to which each candidate meets them. You may also find that some products provide features that you never thought of, but clearly will be of value to you. These should also be noted.

In addition to the software features, the interactions you have with the vendor during this process should provide an excellent indication of the level of support you will likely receive if you purchase the software, and this should always be a key consideration. How responsive was the vendor to your questions? Did the support staff seem knowledgeable and enthusiastic?

In most cases, after experimenting with the software itself, viewing the vendor's test cases, and interacting with the vendor, the best tool for your particular application will become apparent.

GoldSim Technology Group
www.goldsim.com
software@goldsim.com