Algorithm 1.1.1 If done well, a typical discrete-event simulation model will be developed consistent with the following six steps. Steps (2) through (6) are typically iterated, perhaps many times, until a (hopefully) valid computational model, a computer program, has been developed.

(1) Determine the *goals* and *objectives* of the analysis once a system of interest has been identified. These goals and objectives are often phrased as simple Boolean decisions (e.g., should an additional queuing network service node be added) or numeric decisions (e.g., how many parallel servers are necessary to provide satisfactory performance in a multi-server queuing system). Without specific goals and objectives, the remaining steps lack meaning.

(2) Build a *conceptual* model of the system based on (1). What are the *state variables*, how are they interrelated and to what extent are they dynamic? How comprehensive should the model be? Which state variables are important; which have such a negligible effect that they can be ignored? This is an intellectually challenging but rewarding activity that should not be avoided just because it is hard to do.

(3) Convert the conceptual model into a *specification* model. If this step is done well, the remaining steps are made much easier. If instead this step is done poorly (or not at all) the remaining steps are probably a waste of time. This step typically involves collecting and statistically analyzing data to provide the input models that drive the simulation. In the absence of such data, the input models must be constructed in an ad hoc manner using stochastic models believed to be representative.

(4) Turn the specification model into a *computational* model, a computer program. At this point, a fundamental choice must be made — to use a general-purpose programming language or a special-purpose simulation language. For some this is a religious issue not subject to rational debate.

(5) *Verify.* As with all computer programs, the computational model should be consistent with the specification model — did we implement the computational model correctly? This verification step is not the same as the next step.

(6) *Validate.* Is the computational model consistent with the system being analyzed — did we build the right model? Because the purpose of simulation is insight, some (including the authors) would argue that the *act* of developing the discrete-event simulation model — steps (2), (3), and (4) — is frequently as important as the tangible *product.* However, given the blind faith many people place in any computer generated output the validity of a discrete-event simulation model is always fundamentally important. One popular non-statistical, Turing-like technique for model validation is to place actual system output alongside similarly formatted output from the computational model. This output is then examined by an expert familiar with the system. Model validation is indicated if the expert is not able to determine which is the model output and which is the real thing. Interactive computer graphics (animation) can be very valuable during the verification and validation steps.

Example 1.1.1 The following *machine shop* model helps illustrate the six steps in Algorithm 1.1.1. A new machine shop has 150 identical machines; each operates continuously, 8 hours per day, 250 days per year until failure. Each machine operates independently of all the others. As machines fail they are repaired, in the order in which they fail, by a service technician. As soon as a failed machine is repaired, it is put back into operation. Each machine produces a net income of \$20 per hour of operation. All service technicians are hired at once, for 2 years, at the beginning of the 2-year period with an annual salary expense of \$52,000. Because of vacations, each service technician only works 230 8-hour days per year. By agreement, vacations are coordinated to maximize the number of service technicians on duty each day. How many service technicians should be hired?

(1) The objective seems clear — to find the number of service technicians for which the profit is maximized. One extreme solution is to hire one technician for each machine; this produces a huge service technician overhead but maximizes income by minimizing the amount of machine down-time. The other extreme solution is to hire just one technician; this minimizes overhead at the potential expense of large down-times and associated loss of income. In this case, neither extreme is close to optimal for typical failure and repair times.

(2) A reasonable conceptual model for this system can be expressed in terms of the state of each machine (failed or operational) and each service technician (busy or idle). These state variables provide a high-level description of the system at any time.

(3) To develop a specification model, more information is needed. Machine failures are random events; what is known (or can be assumed) about the time between failures for these machines? The time to repair a machine is also random; what, for example, is the distribution of the repair time? In addition, to develop the associated specification model some systematic method must be devised to simulate the time evolution of the system state variables.

(4) The computational model will likely include a simulation clock data structure to keep track of the current simulation time, a queue of failed machines and a queue of available service technicians. Also, to characterize the performance of the system, there will be statistics gathering data structures and associated procedures. The primary statistic of interest here is the total profit associated with the machine shop.

(5) The computational model must be verified, usually by extensive testing. Verification is a software engineering activity made easier if the model is developed in a contemporary programming environment.

(6) The validation step is used to see if the verified computational model is a reasonable approximation of the machine shop. If the machine shop is already operational, the basis for comparison is clear. If, however, the machine shop is not yet operational, validation is based primarily on consistency checks. If the number of technicians is increased, does the time-averaged number of failed machines go down; if the average service time is increased, does the time-averaged number of failed machines go up?

System Diagrams

Particularly at the conceptual level, the process of model development can be facilitated by drawing system diagrams. Indeed, when asked to explain a system, our experience is that, instinctively, many people begin by drawing a system diagram. For example, consider this system diagram of the machine shop model in Example 1.1.1.

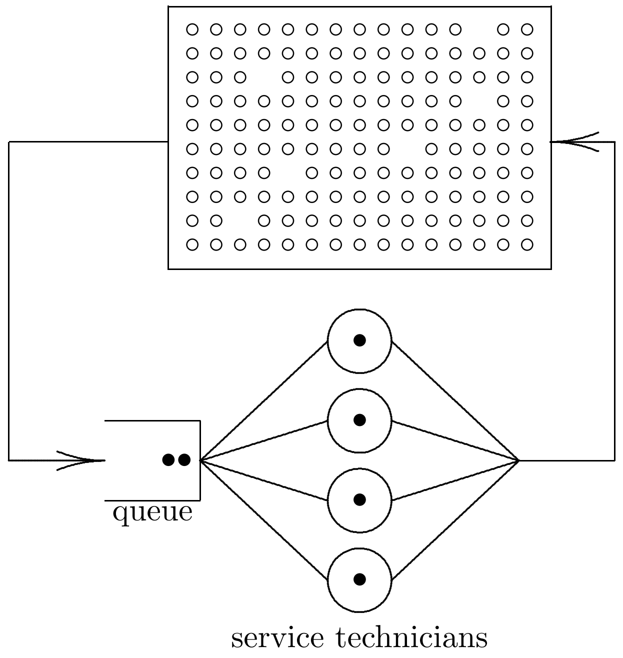

Figure 1.1.2. Machine shop system diagram.

The box at the top of Figure 1.1.2 represents the pool of machines. The composite object at the bottom of the figure represents the four service technicians and an associated single queue. Operational machines are denoted with a ◦ and broken machines with a •. Conceptually, as machines break they change their state from operational (◦) to broken (•) and move along the arc on the left from the box at the top of the figure to the queue at the bottom of the figure. From the queue, a broken machine begins to be repaired as a service technician becomes available. As each broken machine is repaired, its state is changed to operational and the machine moves along the arc on the right, back to the pool of operational machines.*

As time evolves, there is a continual counter-clockwise circulation of machines from the pool at the top of Figure 1.1.2 to the service technicians at the bottom of the figure, and then back again. At the "snapshot" instant illustrated, there are six broken machines; four of these are being repaired and the other two are waiting in the queue for a service technician to become available.

* The movement of the machines to the servers is conceptual, as is the queue. In practice, the servers would move to the machines and there would not be a physical queue of broken machines.

In general, the application of Algorithm 1.1.1 should be guided by the following observations.

- Throughout the development process, the operative principle should always be to make every discrete-event simulation model as simple as possible, but never simpler. The goal is to capture only the relevant characteristics of the system. The dual temptations of (1) ignoring relevant characteristics or (2) including characteristics that are extraneous to the goals of the model, should be avoided.
- The actual development of a complex discrete-event simulation model will not be as sequential as Algorithm 1.1.1 suggests, particularly if the development is a team activity in which case some steps will surely be worked in parallel. The different characteristics of each step should always be kept clearly in mind avoiding, for example, the natural temptation to merge steps (5) and (6).
- There is an unfortunate tendency on the part of many to largely skip over steps (1), (2), and (3), jumping rapidly to step (4). Skipping these first three steps is an approach to discrete-event simulation virtually certain to produce large, inefficient, unstructured computational models that cannot be validated. Discrete-event simulation models should *not* be developed by those who like to think a little and then program a lot.

1.1.3 SIMULATION STUDIES

Algorithm 1.1.2 Following the successful application of Algorithm 1.1.1, use of the resulting computational model (computer program) involves the following steps.

(7) Design the simulation experiments. This is not as easy as it may seem. If there are a significant number of system parameters, each with several possible values of interest, then the combinatoric possibilities to be studied make this step a real challenge.

(8) Make production runs. The runs should be made systematically, with the value of all initial conditions and input parameters recorded along with the corresponding statistical output.

(9) Analyze the simulation results. The analysis of the simulation output is statistical in nature because discrete-event simulation models have stochastic (random) components. The most common statistical analysis tools (means, standard deviations, percentiles, histograms, correlations, etc.) will be developed in later chapters.

(10) Make decisions. Hopefully the results of step (9) will lead to decisions that result in actions taken. If so, the extent to which the computational model correctly predicted the outcome of these actions is always of great interest, particularly if the model is to be further refined in the future.

(11) Document the results. If you really did gain insight, summarize it in terms of specific observations and conjectures. If not, why did you fail? Good documentation facilitates the development (or avoidance) of subsequent similar system models.

Example 1.1.2 As a continuation of Example 1.1.1, consider the application of Algorithm 1.1.2 to a verified and validated machine shop model.

(7) Since the objective of the model is to determine the optimal number of service technicians to hire to maximize profit, the number of technicians is the primary system parameter to be varied from one simulation run to the next. Other issues also contribute to the design of the simulation experiments. What are the initial conditions for the model (e.g., are all machines initially operational)? For a fixed number of service technicians, how many replications are required to reduce the natural sampling variability in the output statistics to an acceptable level?

(8) If many production runs are made, management of the output results becomes an issue. A discrete-event simulation study can produce *a lot* of output files which consume large amounts of disk space if not properly managed. Avoid the temptation to archive "raw data" (e.g., a detailed time history of simulated machine failures). If this kind of data is needed in the future, it can always be reproduced. Indeed, the ability to reproduce previous results *exactly* is an important feature which distinguishes discrete-event simulation from other, more traditional, experimental sciences.

(9) The statistical analysis of simulation output often is more difficult than classical statistical analysis, where observations are assumed to be *independent.* In particular, time-sequenced simulation-generated observations are often correlated with one another, making the analysis of such data a challenge. If the current number of failed machines is observed each hour, for example, consecutive observations will be found to be significantly positively correlated. A statistical analysis of these observations based on the (false) assumption of independence may produce erroneous conclusions.

(10) For this example, a graphical display of profit versus the number of service technicians yields both the optimal number of technicians and a measure of how sensitive the profit is to variations about this optimal number. In this way a policy decision can be made. Provided this decision does not violate any external constraints, such as labor union rules, the policy should be implemented.

(11) Documentation of the machine shop model would include a system diagram, explanations of assumptions made about machine failure rates and service repair rates, a description of the specification model, software for the computational model, tables and figures of output, and a description of the output analysis.

Insight

An important benefit of developing and using a discrete-event simulation model is that valuable insight is acquired. As conceptual models are formulated, computational models developed and output data analyzed, subtle system features and component interactions may be discovered that would not have been noticed otherwise. The systematic application of Algorithms 1.1.1 and 1.1.2 can result in better actions taken due to insight gained by an increased understanding of how the system operates.