

Chapter 2

The Modeling and Simulation Life Cycle Process

Margaret L. Loper

2.1 Introduction

A good way to understand modeling and simulation (M&S) is to look at the process through which models and simulations are developed. There are different M&S life cycles described in the literature (Sargent 1982; Kreutzer 1986; Balci and Nance 1987; Balci 2012). Despite emphasizing different aspects, most represent similar concepts or steps. The process shown in Fig. 2.1 captures the basic ideas represented in most processes.

In a simulation study, there is the person who has a problem that needs to be solved (the client) and the person or group that will solve the problem by developing a simulation model (the simulation analyst). We use client and simulation analyst in the following discussion of the process.

2.2 Steps in the Process

2.2.1 Establish Purpose and Scope

Every simulation study begins with a statement of the problem. Without an understanding of the objectives to be addressed, it is likely the study will not be successful (Law and Kelton 1999). If the client provides the problem statement (i.e., the problem they want to answer with the simulation), the simulation analyst must insure that it is clearly understood. If the simulation analyst prepares the problem statement, it is important that the client understands and agrees with the problem formulation. A good approach is for the simulation analyst to develop a set

M. L. Loper (✉)
Georgia Tech Research Institute, Atlanta, GA, USA
e-mail: margaret.loper@gtri.gatech.edu

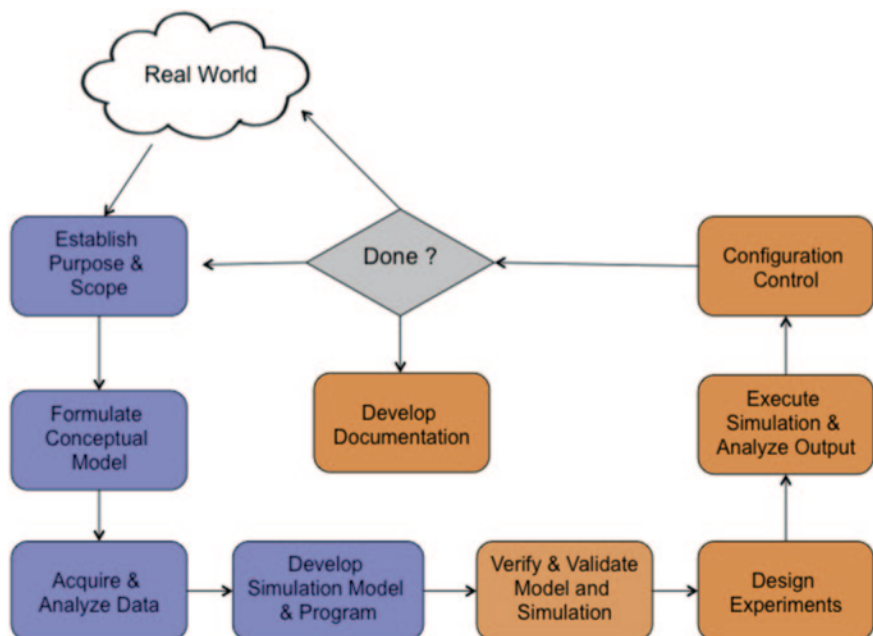


Fig. 2.1 Modeling and simulation process. (Adapted from Benjamin 2006)

of assumptions about the real-world system associated with the problem statement, and make sure that the client agrees with this problem definition. Even if this occurs, it is possible that the problem will need to be reformulated as the simulation study progresses, due to new information and requirements.

The simulation objectives are the questions to be answered by the simulation. The questions could be described as different scenarios that will be investigated. The simulation project plan is a document that includes how much time will be required, people that will be used, hardware and software requirements if the client wants to run the model and conduct the analysis, stages in the investigation, output at each stage, cost of the study and billing procedures, if any. In other words, the project plan documents how you are going to run the simulation project by itself. The amount of detail needed in the project plan should be equivalent to the size of the project, e.g., a large, complex simulation would need a very detailed project plan.

To illustrate how the M&S process works, we will use an example of a full-service gas station (Birta and Arbez 2010). The station has two islands and four service lanes, as shown in Fig. 2.2. Depending on time of day, one or two attendants serve customers. Significant portions of the customers drive vans and light trucks, which have larger gas tanks and take more time to fill. Drivers of passenger cars wait longer for service behind the vans or light trucks, which leads to complaints.

The management of a full-service gas station is considering restricting the vans and light trucks to two of the lanes to improve the flow of vehicles. They want to use M&S to answer the question—will customer service time for passenger cars improve by making this change?

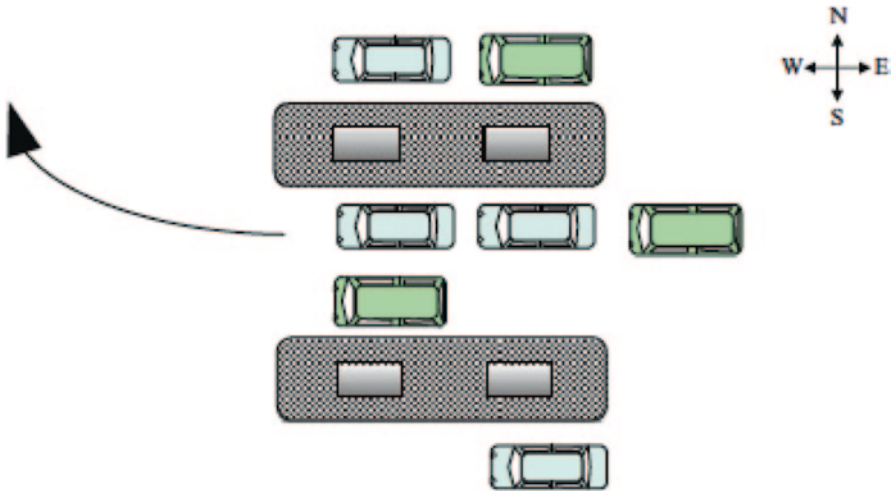


Fig. 2.2 Full service gas station. (Birta and Arbez 2010)

2.2.2 Formulate the Conceptual Model

The conceptual model is an abstraction of the real-world system under investigation. A simulation conceptual model is a living document that grows from an informal description to a formal description and serves to communicate between the diverse groups participating in the simulation's development. It describes what is to be represented, the assumptions limiting those representations, and other capabilities (e.g., data) needed to satisfy the user's requirements. It is commonly recommended that you start with a simple model, adding detail and complexity as needed until you reach the right representation for your problem.

An informal conceptual model may be written using natural language and contain assumptions about what you are or are not representing in your model. An informal model can help clients and simulation analysts understand the basic outline of the model, from their perspectives, on how the real world is represented in the model. A formal conceptual model is an unambiguous description of model structure. It should consist of mathematical and logical relationships describing the components and the structure of the system. It is used as an aid to detect omissions and inconsistencies and resolve ambiguities inherent in informal models, and used by software developers to develop code for the computational model. Software implementation is not discussed in the formal conceptual model.

The first step in the gas station example is to simplify the problem by making assumptions. For example, modeling the attendant's mood or physical characteristics is not important to answering the question of how to improve service time for passenger cars. Therefore, only a simple representation of attendant behavior might be needed. Similarly, assumptions about the importance of road geometry, vehicle dynamics, or weather conditions will also need to be made. In addition to

these types of assumptions, behavioral information is needed to better define model components. For example, the simplified attendant's behavior might be represented by three possible states: pumping gas, taking payment or idle; and vehicle dynamics might be simplified to three characteristics: type of vehicle, size of gas tank, and the location of the gas tank cap (left vs. right). These types of decisions are made based on the purpose of the model and the questions the client would like the simulation analyst to answer. The conceptual model is the place where this type of information is documented; it is a living document and will evolve over time, as the problem is better understood.

2.2.3 Acquire and Analyze Data

Having good data to drive a model is just as important as having sound model logic and structure. Simulation analysts must decide what data are needed, what data are available and whether it is pertinent, whether existing data are appropriate for the required purpose, and how to gather the data. Even though data collection is identified as a separate step in the development process, constructing the conceptual model occurs while data collection is taking place. In a real world simulation study, the gathering and evaluation of input data is very time-consuming and difficult; a significant amount of time used in the study is often consumed by this task.

Regardless of the method used to collect the data, the decision of how much to collect is a trade-off between cost and accuracy. There are several potential sources of data: historical records, observational data, similar systems, operator estimates, vendor's claims, designer estimates, and theoretical considerations. Systems differ with regard to how much data are available to populate the system database. Data rich refer to systems where data are abundant from prior experiments or obtained from measurements. Data poor refer to systems where meager amounts of historical data or low-quality data exist. In some cases, it is impossible to acquire better data (e.g., combat), in others it is too expensive to collect (e.g., topography and vegetation of a forest).

In the gas station example, the simulation analyst will need to collect a variety of data for the model. Some of the data will be used as input and some of it will be used in the algorithms or equations used inside the simulation. Let us consider traffic flow. How many vehicles will come into the gas station? Will the number be constant or will it change with the time of day or the day of the week? What percentage of vehicles are passenger cars versus trucks or light vans? Data need to be collected to answer all of these questions. In addition to traffic flow, we will need data to represent the attendant's behavior. How much time does it take for a particular vehicle to have its gas tank filled? How long does it take for the attendant to process a customer's payment? Is it possible for customers to take longer in one of these steps, maybe because they exited the vehicle to go to the restroom? The simulation analyst can choose to gather empirical data (e.g., spend the day at the gas station and count cars) or use probability distributions, based on real data, to represent

these aspects of the model. Some of the data will be used as input and can be varied during the simulation study (e.g., the number of cars visiting the gas station) and some of the data may be used internal to the simulation as part of an equation and be considered hardwired (e.g., size of the gas tank for each type of vehicle). The data and any assumptions made during the collection process should be documented with the conceptual model.

2.2.4 Develop Simulation Model and Program

In this step, the conceptual model is coded into a computer recognizable form, an operational model. Translating the model into computer code and then into an executable involves selecting the most appropriate simulation methodology (e.g., discrete event, continuous, agents, or system dynamics which are defined and discussed in later chapters) and an appropriate computer implementation (e.g., programming or simulation language). Simulation analysts may choose to develop the simulation using a programming language (e.g., C, Java, FORTRAN) or use a simulation language (e.g., MATLAB®, Simio, NetLogo, STELLA®¹). Simulation languages are higher-level software packages than programming languages. They are usually custom made for more specific purposes and come with additional tools to facilitate their use. Programming languages are low-level software packages, meaning the computer code using these languages is converted into computer code using a compiler. Programming languages are used to write custom-made software. For example, it will take a computer programmer a lot longer to put together an agent-based simulation using a programming language as opposed to an agent-based framework like NetLogo. NetLogo has many internal capabilities to facilitate creating a simulation, whereas everything has to be developed from scratch using a programming language. However, simulations using programming languages can have advantages such as customization and faster execution.

There are different ways a simulation analyst might want to represent traffic flow for the gas station, depending on the purpose of the model and questions the client wants answered. For example, traffic can be represented in a simulation as a queueing model, as a fluid flow model, or as cellular automata. Each of these modeling methods has advantages and disadvantages and focus on different types of mathematics and behavior representation. There is no single right way to select a modeling method for a simulation, but once selected there are well-defined ways to do the implementation. Simulation languages usually implement a selected subset of modeling methods; so developing the simulation is about combining the appropriate techniques and libraries needed (e.g., discrete-event simulation languages, such as NetLogo, all represent queueing models). However, simulation languages may not have the specific capability required; so a simulation analyst may need to

¹ MATLAB® <http://www.mathworks.com/products/matlab/>; Simio <http://www.simio.com/>; NetLogo <https://ccl.northwestern.edu/netlogo/>; STELLA® <http://www.iseesystems.com>.

use a computer language to custom code the required algorithms. It is important to recognize that the simulation implementation is closely related to the conceptual model. In fact, the conceptual model should include enough information about the gas station (i.e., vehicles, attendant behavior, traffic flow) that the simulation implementer has all the information they need to choose the appropriate language and modeling approaches.

2.2.5 Verify and Validate the Model and Simulation

Verification is the process of determining that a model or simulation implementation and its associated data accurately represent the developer's conceptual description and specifications. It is often summarized as "Did we build the model right?" Verification should be continuing process; the simulation analyst should not wait until the entire model is complete to begin the verification process. Using interactive tools, such as a debugger, can aid to the verification process.

Validation is process of determining the degree to which a model or simulation and its associated data are an accurate representation of the real world. It is often summarized as "Did we build the right model?" Can the model be substituted for the real system for the purposes of experimentation? If there is an existing system, an ideal way to validate the model is to compare its output to that of the existing system. There are many methods for performing validation (Balci 1997).

As part of the verification and validation (V&V) process, the simulation analyst will need to verify the requirements of the problem stated by the client, validate the conceptual model, verify the design and implementation of the simulation, and then validate the results of the simulation. In other words, even though V&V is depicted in Fig. 2.1 as occurring halfway through the life cycle process, it actually occurs at every step of the process.

In the gas station example, the simulation analyst will work with the management of the gas station to verify they understand the question to be answered, the purpose of the model, and any requirements defined by the client. In other words, the simulation analysts should make sure that the requirements are associated with changing the flow of vehicles with the two islands at the gas station, and not about adding a third island. Clients and simulation analysts often use different words for similar concepts, which can lead to a misunderstanding of requirements. Verifying requirements ensures the client and analyst mutually understand the purpose of the model.

When developing the conceptual model, the simulation analyst needs to examine the assumptions, behavior, and data against the requirements of the problem. For example, if the problem is focused on improving customer service time for passenger cars by changing flow inside the gas station, then there is likely no reason to be worried about representing weather conditions. Varying the number of vehicles entering the gas station can easily represent the general idea of fewer cars visiting the gas station on bad weather days. Similarly, if the conceptual model did not

include the concepts of vehicles, traffic flow and attendants, there would be a real disconnect with the requirements of the problem.

In the verification process, the analyst needs to ensure that the models have been implemented correctly into the simulation. The question being answered in the verification is: “are the equations being solved correctly?” In real-life projects, sometimes an independent team is assembled to go through the software code and examine the results to make sure the simulation is working. In complex projects, it is not feasible to verify the code line by line. In this case, a second simulation might be built by an independent team to verify the simulation. If the models have been implemented correctly, the results of the two independent simulations should match, otherwise the analyst needs to track down the cause of the discrepancy. When verifying the design and implementation of the gas station simulation, the simulation analyst needs to make sure that the simulation runs and does not produce errors. If a simulation language is used for the implementation, the algorithms, formalisms, equations, etc. have already been verified. However, the specific use of data and combination of the equations will still need to be tested to ensure that the logic of the simulation works and is “bug free.” For example, if the simulation were computing average traffic flow into the gas station, a divide by zero would cause a “runtime error” and cause the simulation program to crash.

Once the simulation of the gas station is running, the analyst will need to validate the results of the simulation. The simulation should be executed using the “as-is” configuration of the traffic flow in the gas station (i.e., first come, first served in any lane of the two islands). The simulation results for customer service time of passenger cars should closely match what the service time is in real life. For example, the number of cars entering the gas station is a function of bad weather. To validate the model, the analyst needs to gather data on the percentage of traffic reduction during bad weather, and making sure the model reflects this as best as possible. To validate the simulation results does require knowing what the behavior of the real system is and how it operates. This is part of data collection: to gather this type of information. If the simulation results are not the same (or similar, within some error bound) as the real gas station, then the simulation analysts and client will not be able to trust the results of the simulation when the traffic flow (i.e., passenger cars one island, vans and light trucks one island) is changed.

2.2.6 Design Experiments

For each scenario that is to be simulated, decisions need to be made concerning the length of the simulation run, the number of runs (also called replications), and the manner of initialization that is required. One approach to conducting experiments is to vary the input data for each simulation run and observe the behavior of the system. If there are only a few input parameters, this approach will work. However, if the simulation has tens or hundreds of input parameters, conducting a run for each combination of the parameters can be unmanageable. In this case, a formal process

called design of experiments (DoE) can be used to better define which runs are needed. DoE is a statistical approach to experimental design (described in a later chapter) that can help minimize the number of simulation runs needed to understand the behavior of a system under investigation. The simulation analysts may also want to use a technique called Monte Carlo analysis (described in a later chapter) that relies on repeated random sampling to compute the results of the simulation.

Defining the experiments for the gas station model should be considered while developing the conceptual model. The reason is that the simulation analyst needs to make sure the right assumptions, behaviors, and data are collected and included in the simulation in order to support the necessary experiments. A set of experiments an analyst might run for the gas station management would be to vary the number of vehicles that come into the station. This could include increasing the number of passenger cars, light trucks, and vans separately, or increasing them in different combinations.

To illustrate the importance of thinking through experiments during the conceptual model development, what would happen if you were presenting the results of these runs to the client (i.e., gas station management), and they ask you whether the results change if Sue is working as an attendant versus Bob? Since the behavior of the attendant was simplified and does not include personal attributes of specific people, the simulation cannot answer that question directly. The simulation analyst will need to understand the difference between Sue and Bob's performance and how that can be included in the simulation as the service time of the attendant, and then make changes in the simulation to include that as an input parameter and not hardwired into the equations for the attendant. The changes needed may not be that difficult, but it requires time and resources to change the conceptual model, collect data, make the changes to the simulation, do the verification and validation, and run the experiments again. Upon presenting the new results to the gas station management, they ask if the results are different if there is a football game at the stadium down the block. If the simulation does not allow traffic to be input hourly (say it was originally implemented as a probability distributed), then more time and resources will be needed to make the changes. Defining the sets of experiments early in the simulation process can prevent many unnecessary hours or reworking the simulation.

2.2.7 Execute Simulation and Analyze Output

The final runs, in which data are collected for analysis, are known as production runs. They are used to estimate measures of performance (MOP) for the scenarios that are being simulated. MOPs are a measure of a system's performance expressed as some quantifiable features, such as customer service time in the gas station. By analyzing the MOP of the production runs, the simulation analyst determines if additional runs are needed and if any additional scenarios need to be simulated. A scenario is a description of the initial set of conditions and timeline of events used

to evaluate the behavior of the system. For example, you may define one scenario to look at the behavior of the gas station when the attendants are attentive to customers (e.g., they are always standing at the gas pump so service can begin immediately once a vehicle arrives) and when attendants are not attentive to customers (e.g., they are inside waiting for a vehicle to arrive and take several minutes to get to the pump to begin service).

In addition to determining if more runs are needed, analyzing the results could indicate that the simulation needs to be modified. In this case, the simulation analyst should go back to the beginning of the process to understand whether the modifications represent a change to the scope and purpose, or whether it is only a modification to the simulation implementation.

Once the experiments have been defined, the simulation analyst will need to decide how many times to run the simulation for each scenario in order to have confidence in the results. Depending on the amount of uncertainty included in the simulation, the analysts may want to run each experiment multiple times in order to gain more insight into the impact of uncertainties on the overall results. For the gas station, we may decide to run each experiment five times with the same input, and then average the results. It is important for the analyst to examine the simulation results and be on the lookout for outliers. Outliers are data points that are numerically distant from the rest of the data. Outliers need closer inspection to determine why they numerically deviate from the rest of the data. Some outliers expose flaws in the system uncovered by the simulation. In some instances, inputs can conspire in such a way as to break the system. In such a case, the system design needs to be reexamined and necessary modifications made. Such modifications usually make the system more robust. Some outliers expose bugs in the simulation itself. In this case, the simulation bugs need to be identified and fixed and the experiment rerun. In addition to the number of runs, we may discover strange behavior in the results, e.g., on Monday's customer service time for passenger cars doubles. This would indicate a behavior we would want to investigate further to understand if it is an error in the simulation logic or if there is a valid reason for this behavior (e.g., everyone waits until Monday to fill up their car for the week). There are many output analysis techniques that can be used by a simulation analyst to help them analyze the results (Nakayama 2006).

2.2.8 Configuration Control

A simulation package for a real-world project can be made up of thousands to millions of lines of computer code. Some computer simulation packages could be a collection of several individual software packages linked together. What may seem like a small change can lead to large excursions in the behavior of the overall simulation. Thus, it is important to keep control of the package and its contents, known as its "configuration." The "configuration control" is practiced in most, if not all, real-world projects that have a software component. Once it is decided that the simula-

tion works and has passed all the required tests, the configuration control manager preserves the system in its current configuration and assigns a configuration control number to it, so at any point in the future, it is possible to revert back to the last saved configuration control version, in case future revisions do not work. The simulation analyst must never change a parameter in the configuration-controlled version of the program until given the authority and any changes must be documented. Usually, the engineer develops a “working copy” until ready to incorporate and document changes on the configuration-controlled copy. After enough changes are made, the new configuration controlled version is created with a different configuration control designation number to differentiate it from all previous releases. The new configuration-controlled copy is typically identified by a “release” or “version” number or date; for example, “Release 1.6” or “Version 2012_03_17” (March 17, 2012).

2.2.9 Develop Documentation

Documentation is a very important part of the simulation process. If the simulation model is going to be used again by the same or different analysts, it may be necessary to understand how the simulation model operates. Also, if the model is to be modified, this can be greatly facilitated by adequate documentation.

It is important that all the simulation inputs are documented. All the simulation interfaces need to be identified, and the format of all the input data, and their units or dimensions, needs to be specified. The documentation needs to specify if an input file is a regular text file, or in binary format. If the input file is in binary, the documentation needs to explicitly spell out the format of the data byte by byte. If the input file is in regular text format, documentation needs to specify the number of columns, columns headers, how the columns are separated (comma, space, tab, etc.), number of data points, etc.

The result of all the analysis should be reported clearly and concisely. This will enable the client to review the final problem formulation, the alternatives that were addressed, the criterion by which the alternative systems were compared, the results of the experiments, and analyst recommendations, if any.

There are two types of documents we can envision for the gas station model. One is a final report delivered to the gas station management documenting the project. It will include a definition of the requirements and the questions being answered, some aspects of the conceptual model that is understandable to the client, a description of the simulation developed, the experiments conducted, and the simulation results. The requirements for the final report will depend on the specific client, so the simulation analyst will need to clarify how much detail the client wants. The second type of document the simulation analyst will want to develop is a complete set of documentation on the project itself. This will include all aspects of the conceptual model, design and implementation of the simulation, the verification and validation tests, and all of the data collected and analyzed. This is needed to fully understand

Modeling and Simulation in the Systems Engineering
Life Cycle

Core Concepts and Accompanying Lectures

Loper, M.L. (Ed.)

2015, XIX, 410 p. 165 illus., 6 illus. in color., Hardcover

ISBN: 978-1-4471-5633-8