Verification and Validation of Simulation Models

The verification and validation of a simulation model goes a step beyond the kind of testing done for ordinary software systems. For ordinary software systems, testing is conducted to verify that the system generates absolutely predictable outcomes based on test data. *Simulation models have an expected rather than an absolute behavior,* and may have widely differing results depending on configuration and input data. The kind of testing used in development of software systems is used to get a simulation model in functional order, but additional testing is required for verification and validation of the simulation model. In large measure, many of the methods employed are informal subjective comparisons (which one suspects is also the primary testing mechanism of systems software); with added effort, formal statistical testing techniques are also used, but often require laborious acquisition of real data heretofore uncollected.
Verification

Simply put, verification is the task of determining if the implementation of a model has been done correctly. Beyond program debugging, this means that verification data needs to be generated at various points in the model for comparison with expected values. A good example is the output of mean interarrival times at points matching those from stability analysis. Where differences are substantial, in all likelihood, part of the model implementation has an error (e.g., a default delay time is being used because the delay input for a server did not get hooked up). The more difficult issue of how well a distribution that has been used represents an actual data scenario is part of the validation process.
Validation

Validation is the task of determining if the model constructed accurately represents the underlying real system being modeled.

For any simulation model that is to be used in actual application it is very important to validate the model insofar as practicable, since real decisions are going to be made based on the simulation outcomes.

In many cases, there is an existing system that can be captured as part of the simulation model. In this case, the model can be configured to match the characteristics of the existing system and the model outcomes can be compared to outcomes measured from the real system.

Because a simulation model provides a surface “realism”, it is possible to be fooled by the realistic appearance of the simulation. The best defense against this kind of mistake is to employ multiple means of comparing model performance against real data (if available), including statistical testing.
Validation Techniques

Van Horn (Duke, 1971) has a widely cited list of techniques, in increasing order of real cost:

1. In model construction, involve people with domain knowledge, particularly for developing the model elements. Domain experts should also have expert knowledge on where and what input/output data to incorporate. There may be previous work (such as lab experiments) or statistical models from which some model elements can be extracted or developed.

2. For any predefined probability distributions that are used, conduct statistical “goodness-of-fit” tests to real system data (assuming data of reasonable quality exists). It is also advisable to have the model report standard statistical measures for generated data to verify that these correspond to expected values.

3. A “Turing Test” is one where both real and “fake” data are examined to see if it can be determined which is which. Produce a set of model runs and a set of real runs in the same format and present them to people with expert knowledge of the system. Present them with the task of determining which runs are model generated and which are not. If the knowledge experts cannot distinguish which is which with any consistency, then the test is said to fail to demonstrate model inadequacy. If the knowledge experts can distinguish which is which with any degree of accuracy, then the modeler needs to question the knowledge experts to determine what modifications are needed in the model.

4. Conduct statistical analyses comparing model output with real system output. This may require acquisition of system data that is not currently being captured.

5. Once 1-4 are done, deploy the model and repeat 2-4 with newly generated model and system data.

6. Based on simulation outcomes, implement the new or modified system, collect data from the system and repeat 2-4. Unfortunately, this is sometimes the only step taken!
Implementing without Validation

*It is rarely acceptable to implement a simulation model with little or no validation,* because the model may be badly miscast due to ingrained assumptions of the modeler (from experience or otherwise). For example, I may have preconceived ideas of flow processes based on my real world experience, but not understand the ways in which flows may change depending on both materials and external factors. If the model design is constrained by my preconceived ideas (and not well-validated), then it may produce disastrously wrong information if employed with materials outside of my experience. It is easy to visualize scenarios where this kind of thing may happen even when doing high quality modeling work; for example, if we wanted to model flow processes on the surface of the sun.
Model Building

1. The model building process is characterized by
   - observation of the real system
   - determining its main components and how they interact
   - collection of data on system behavior

   There are usually knowledge experts for one or more parts of a system who can provide insight and feedback on the modeling structure employed.

2. Once a system is (more or less) understood, it is possible to describe a conceptual model of the system. The conceptual model usually tries to capture (in English) the component elements of the system and the data characteristics associated with them (note that this is how the Extend modeling assignments are generally presented).

3. The conceptual model should have enough detail to serve as the basis for development of a working simulation model.

   The model building process should be viewed as being iterative; i.e., the initial try at building a working simulation model may send the developer back to the conceptual model and from there back to the real system to better elaborate parts that have not been closely enough described.

   Diagrammatically (as per the course text), the process has the structure

![Diagram of model building process]
Model Verification

In model verification, the answer to the question “does this make sense given what I know about the conceptual model and the real-world system?” is the one which is most important.

Fundamental tactics for model verification include:

1. **Having the implementation presented** to and checked by someone not directly involved in its development.
2. **Tracing the model flow** via flow diagram or otherwise to determine if the model is taking all possible actions in the course of a run (note that this is critical when using components such as Extend’s Select DE Output block (DE.lix)).
3. **Model outputs making sense** for various combinations of possible configurations and input data; it is important in this instance that as much output of expected values as can be stomached should be incorporated into the model, most typically “long-run” statistical measures that can be predicted at various points in the model, such as the mean and standard deviation of time data, item counts, and elapsed time (particularly for catching the common mistake of using different time units in model components).
4. **Dumping data** to see if the values are what they should be at simulation end.
5. **Including explanatory comments** either in block dialogues or in text boxes on Extend’s work plane.
6. **Observing model behavior under animation**, a powerful tool to visually check that model components are behaving as expected; you must be very cognizant that Extend’s animation is strictly sequential, so actions that occur simultaneously are animated one following the other – this may require keeping a careful eye on the system clock.
7. **Utilizing pause and single step features** to interactively probe intermediate system results (and even change some values); note that the Extend Information block (DE.lix) provides the means for capturing data item by item for facilitating this kind of approach.
Conceptual Validation

Model validation is not a completely separate process from model verification and modelers will inevitably conduct elements of both simultaneously.

The term *calibration* is sometimes used to describe the process of iteratively making model adjustments (major or minor) in comparing model behavior and outcomes to those of the real system. It is important to calibrate the model for a base real system case. If there is more than one variation of the real system, the model needs to be calibrated for each. Since the model may be intended for examination of an extended (non-existent) version of the system, there is an underlying assumption the model integrity will not be corrupted by extending it.

Face Validity

Since ultimately a model is usually intended for the use of people knowledgeable of the real system, it is important for the model (both conceptually and as implemented) to appear to be a reasonable representation of the real system. Obviously, it is important to involve these possible users in model development, and they should be asked to evaluate the model as it develops as to its reasonableness. In particular, when model parameters are changed, these types of users can address the question of whether or not the resulting model behavior is “reasonable” and what they would expect in the real world case.

*This kind of testing should be carefully planned*, because the real users will not have a grasp of what constitutes a reasonable set of test cases. Also, it may be prohibitive (time-wise or otherwise) to exhaustively test all possible variations of the configuration. It may also be possible to check model validity for this purpose by using statistical sensitivity tests (assuming enough real system data is on hand).
Validation of Model Assumptions

- **Structural Assumptions**
  These are simply the simplifications and abstractions of how the real system operates, which must be compared to real system behavior

- **Data Assumptions**
  Since the model generates its own data as it executes, the data generation processes are best based on reliable data collected from the real system in actual operation. It is normally possible to statistically compare model generated data with this kind of collected data. It may be necessary to undertake additional data collection to generate reliable data sets for this purpose.

Validation of Input-Output Transformations

Ultimately, the model is expected to be able to accurately predict real system behavior under multiple scenarios. If it cannot, then it will not have been credible for guiding the process of making systematic changes. This can be accomplished simply by demonstrating accurate “prediction of the past” where there is data to support such predictive behavior.