*Proceedings of the 2021 Winter Simulation Conference*
*S. Kim, B. Feng, K. Smith, S. Masoud, Z. Zheng, C. Szabo and M. Loper, eds.*

# A COLLABORATIVE TEST AND DATA-DRIVEN SIMULATION SOLUTION

German Reyes, Ph.D.
Michael Allen
Dayana Cope, Ph.D.

Engineering USA
55 W Monroe Street
Suite 2575
Chicago, IL 60603, USA

## ABSTRACT

A strategic design approach during simulation model development is an increasingly important requirement to support large-scale simulation development. In this case study, we present the importance of adopting two design approaches during the development of simulation models; namely, test-driven development, and data-driven modeling. In addition, in a multi-developer setting, the model architecture played an integral role in enabling collaboration by separating model components while ensuring appropriate interoperability between components and minimized redundancy. A modular design made it possible for developers to focus on specific behavioral aspects of the simulation model, test model behavior in a test environment and integrate all of the architectural pieces into the main master model. The model is automatically generated from data stored in an external database. This modeling approach has proven extremely useful in a setting where multiple developers worked on different model functionalities asynchronously.

## 1 PROBLEM STATEMENT

When multiple developers are working asynchronously within one large-scale simulation model, there is a need for an improved development methodology that is efficient and seamless. This need can be resolved by defining a model architecture where separation of concerns is established, thus allowing each developer to work on their model components in isolation. However, this development methodology brings a new challenge: how can we ensure a module (hereafter referred to as ‘manager’) is able to provide the intended functionality once it is merged with the main simulation model? This challenge is addressed by incorporating a test-driven development approach where test suites are created that test the functionality of the manager, including all identified edge cases. Lastly, one of the lingering challenges in this type of development has to do with the automatic creation of the full simulation model. Since the managers already provide the functionality and control the behavior of the simulation objects (entities, resources, etc.), adopting a data-driven approach has provided us with the capability of creating the simulation objects that encompass the full simulation model.

## 2 MODEL ARCHITECTURE

We developed a robust model architecture in which model features can be broken down into smaller managers that are decoupled by design and provide for asynchronous development of manager libraries. Managers have been designed to communicate with other managers via interface methods, similar to APIs. In addition, managers can update their states based on events published by other managers. In other words, managers can also be configured to be publishers, subscribers or both in a Pub/Sub design pattern implementation. This level of decoupling allows the managers to be individually and independently tested by the use of test suites.

## 3 TEST-DRIVEN DEVELOPMENT / AUTOMATED MODEL VERIFICATION

Significant time can be spent in debugging simulation models, either due to bugs identified in the code base or due to logical errors. Developers may spend countless hours trying to replicate bugs reported by the integration engineers before they can be addressed and released as bug fixes. Having a test suite prevents many of these issues. In this case study, we present the development and use of a test suite that involves the creation of mock managers that mimic the behavior of other managers at a basic level; notably, the mock managers can be as complex as necessary. Under this type of development, engineers write the tests that need to be passed for a given functionality; then, the feature is developed for that specific manager, and tests are run against it to ensure that all tests pass. Passing all tests defined in the test suite is a pre-requisite before new manager libraries can be released and integrated with the main model. A further benefit is the automation of model verification, ensuring changes and updates do not break existing functionality and assuring the quality of the finished model.

## 4 DATA-DRIVEN MODEL

A data-driven model is an effective way to automatically generate a new model configuration by simply changing parameters in an external database. This is of critical importance in large-scale models, where it may be of interest to assess the impact of local changes to the entire system. For example, in a manufacturing site, we may consider changes such as moving work centers or shops to new locations and/or replacing aging equipment with new equipment that has different capabilities. These types of experiments require additional manual development when performed using a conventional model development approach. In addition, experimentation may increase the difficulty of maintaining these models. A data-driven approach removes much of the complexity and allows for greater flexibility when creating a new configuration of the model.

A critical benefit gained from a data-driven approach is the ability of storing the model data in an external database rather than the model itself. A database can more effectively handle tests on the data and prevents modelers from loading incorrect data into the model that may result in errors during a simulation run. Typical tests include checking to see if data types are correct, that there is no missing data, and that data conforms to specific rules. For example, when dealing with probabilities, the sum of the probabilities in a given group cannot exceed 1, or that there are no negative probability values. Performing these sanity checks while the data is in the database allows modelers to diagnose any potential issues that may affect the simulation run. Once all sanity checks have been satisfied, the data is ready to be consumed by the model, and the simulation model is then capable of self-configuration upon initialization.

## 5 DATABASE USER INTERFACE AND SCHEMA CHANGES

As modelers gain more knowledge about the processes being simulated, it becomes necessary to have a means of updating the schema, constraints, or other key aspects of the database. It is often required to add or drop fields from existing tables, or even to create or delete tables in the database. Although these changes can be easily done manually, it becomes increasingly difficult to track what changes have been applied. In this case study, we present the use of a web-based UI framework (such as the Play framework) that has proved to be an excellent solution when managing automatic schema changes from a web UI via evolution scripts. Having a web UI not only provides the aforementioned capability but also provides users with enhanced views of the database contents without the need to continually write queries. Furthermore, a web UI enables end users to not only view the information contained in the database, but removes the need-to-know SQL to perform complex queries. The web UI can be used to edit or add new data to the database either by the use of editing forms, or the use of bulk processing. Lastly, view pages can be easily configured to display KPIs of interest for simulation results that are written back to the database after each simulation run.