Test Automation with the ADvantage Simulation Framework

Complex systems have immense testing requirements and therefore test automation becomes a key method for reducing cost and improving project efficiency. Test automation for real-time simulation is a subject with many aspects. The purpose of this paper is to examine this test automation and shed light into the techniques used to perform highly repeatable testing.
Test Automation with the ADvantage Simulation Framework

Introduction

The increasing complexity of aerospace, defense, and automotive development projects is caused not only by more complex technical designs, new features, and network-distributed electronic systems, but also by complex commercial relationships. Lead system integrators (LSI) or OEMs divide up the design of a particular system into manageable subsystems. These subsystems are assigned to engineering groups and outsource suppliers. Suppliers often subcontract software coding and unit test to lower-tier suppliers. When in operation, multiple electronic units, developed by various suppliers, must perform high-speed, time-sensitive communication with many adjacent subsystems. During project integration, identifying who is responsible for a particular bug can be an enormous and costly challenge with the potential to descend into a counterproductive “blame game”.

Real-time simulation has become a mainstay for complex development projects. Upon completion of a first-generation subsystem prototype, each development group (in-house or outsourced) connects the prototype to a real-time simulation of the adjacent subsystems. The real-time simulation enables a functional test matrix to be performed on the prototype subsystem.

The test matrix defines:
- stimulus to be applied to the subsystem under test, and
- the expected response

Test matrix entries may include static conditions applied to the subsystem, profiles of time-based stimulus, and profiles of event-based stimulus. Test matrices are first developed using desktop simulation and evolve through each stage of subsystem development and integration. Engineering groups and subsystem suppliers share test matrices with those groups developing adjacent subsystems. These shared test matrices are used to communicate interface requirements and test criteria. Ideally, the LSI recognizes the importance of shared test matrices, adds this to the contract requirements, and includes test matrix exchange milestones in the project schedule.

As system scope and complexity grow, the complete system test matrix tends to grow exponentially. Complex systems have immense testing requirements and therefore test automation becomes a key method for reducing cost and improving project efficiency. Test automation for real-time simulation is a subject with many aspects. The purpose of this article is to examine this test automation and shed light on the techniques used to perform highly repeatable testing.
Categories of Test Automation

Test automation for real-time simulation can be divided into three categories, each with a particular list of functional and technical requirements.

These categories include:
- Enterprise Test Automation
- Simulation Host Scripting
- Real-Time Scripting

<table>
<thead>
<tr>
<th>Real-Time Simulation Lab Architecture</th>
<th>Test Automation Category</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="test-automation-enterprise.png" alt="Diagram of Enterprise Test Automation" /></td>
<td>Enterprise Test Automation</td>
</tr>
<tr>
<td>- Configure and initialize human factors systems</td>
<td></td>
</tr>
<tr>
<td>- Configure and initialize scene-generation systems</td>
<td></td>
</tr>
<tr>
<td>- Obtain revision controlled models, test scripts, and configuration file from the Source Database</td>
<td></td>
</tr>
<tr>
<td>- Save test results to the configuration controlled Results Database</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Real-Time Simulation Host</th>
<th>Simulation Host Scripting</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="simulation-host-scripting.png" alt="Diagram of Simulation Host Scripting" /></td>
<td>Automate non-real-time simulation tasks (ADvantageVt)</td>
</tr>
<tr>
<td>- Initialize simulation models</td>
<td></td>
</tr>
<tr>
<td>- Load real-time scripts</td>
<td></td>
</tr>
<tr>
<td>- Configure datalogging</td>
<td></td>
</tr>
<tr>
<td>- Group common tasks using subroutines and functions</td>
<td></td>
</tr>
<tr>
<td>- Configure real-time plotting</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Real-Time Simulated Subsystems</th>
<th>Real-Time Scripting</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="real-time-scripting.png" alt="Diagram of Real-Time Scripting" /></td>
<td>Simulation stimulus applied with microsecond determinism</td>
</tr>
<tr>
<td>- Static conditions</td>
<td></td>
</tr>
<tr>
<td>- Time-based stimulus profiles</td>
<td></td>
</tr>
<tr>
<td>- Event-based stimulus profiles</td>
<td></td>
</tr>
<tr>
<td>- Time- and event-based stimulus profiles</td>
<td></td>
</tr>
<tr>
<td>- Triggered datalogging</td>
<td></td>
</tr>
<tr>
<td>- Data playback</td>
<td></td>
</tr>
<tr>
<td>- Real-time expression execution</td>
<td></td>
</tr>
</tbody>
</table>
Enterprise Test Automation

Advanced simulation labs typically rely not only on one or more real-time simulation computers, but also on other computer systems which may include:

- Cockpit / navigation / driver displays
- Scene generation
- Revision control database systems for managing files including simulation model source, test scripts, and configuration files
- Results databases for managing test results

Enterprise test automation is used to automate all aspects of a complex simulation facility. For a given test:

- All source files are checked out of a revision control database
- Display systems are initialized, the real-time simulation(s) is loaded and initial conditions are set, the simulation is started
- The test performs one or more specific maneuvers
- The test completes, results are recorded and analyzed, and
- The results are checked into a results database in the form of an automatically generated report

Enterprise test automation is used to minimize the human effort required to complete all aspects of a test. Ideally, each test is encapsulated with a single entry point or file. Double-clicking the file starts the enterprise test automation tool and allows the user to kick things off with the press of a button. ADvantage™ was developed for the purpose of performing enterprise test automation. Alternatively, many organizations develop their own Visual Basic or C++ applications to provide enterprise test automation functionality that plugs into the ADvantage simulation framework.
ADvantage™ leverages Windows ActiveX/COM and .NET automation capabilities to automate each component in the real-time simulation lab. In addition, ADvantage™ is tightly integrated with ADvantageVI and minimizes the effort required to perform common real-time simulation setup and execution tasks. ADvantage™ includes a sophisticated XML-based report generation engine. Each automated system within the lab reports back to the report database as the test progresses. Upon test completion, the report is generated with the press of a button or is automatically generated as part of the test.

### Simulation Host Scripting

The test-to-test variance exhibited by the host computer does not get in the way of truly deterministic and repeatable real-time simulation and test.

Advanced real-time simulation systems, such as the ADvantage simulation framework, employ a “simulation host / real-time target” architecture to provide a complete simulation system that is both:

- Fully deterministic (target) and;
- Allows for asynchronous user interaction and display (host).

The simulation host provides a broad range of functionality for interacting with the real-time simulator. The host computer uses a general purpose operating system such as Microsoft Windows. Not being a preemptive real-time operating system, Windows does not promise time-based repeatability. This manifests itself as test jitter and results in timing variance from test to test. However, the test-to-test variance exhibited by the host computer does not get in the way of truly deterministic and repeatable real-time simulation and test. Tests are made to be completely repeatable by executing all activities, required to be repeatable, directly on the real-time simulator (rtX or RTS) and allowing the non-repeatable user interaction activities to occur back at the host computer.

The ADvantage simulation framework includes the run-time application: ADvantageVI. ADvantageVI offers a depth of features including a powerful command-line scripting language used for simulation host scripting. ADvantageVI scripting may be used automate all the features available to the graphical user interface. Commonly performed ADvantageVI scripting activities include:

- Loading and initializing simulation models
- Building up and loading real-time scripts
- Configuring datalogging for visual display and post-run analysis
- Configuring plots
- Looping and branching through repeatable test scenarios
As embedded systems progress through real-time integration testing, problems are found and solved. A considerable challenge when dealing with highly distributed, networked embedded systems is to ensure that a bug fixed in one subsystem does not result in a bug created in another subsystem. Repeatable, automated testing is an important tool for isolating integration problems and eliminating them effectively. When a problem is found, the failed test case(s) and all successfully run test cases are noted. Changes are then made to the design or implementation for the purpose of solving the problem and these changes are integrated back into the system. Upon implementing the fix, the problematic test case(s) is executed to ensure that the problem is indeed solved. Next the test cases that previously passed are executed. If these previously successful test cases remain successful, then integration may continue. If one or more of the previously successful test cases fails then the problem has been pushed into an adjacent subsystem and not solved. This approach to integration testing is exhaustive and reliable but is also very time consuming; hence the need for test automation.

When performing exhaustive integration testing, test repeatability is of the utmost importance. Poor test repeatability may result in a test passing today and failing tomorrow under identical conditions. If this occurs, the complete integration test process becomes unreliable and inefficient. In order to get maximum test repeatability with real-time simulation, test scripting must be executed on the real-time target in lock step with the simulation models.

Traditionally, real-time test maneuvers were implemented by adding maneuver elements to the simulation models. For example, a maneuver would be added to a Simulink-based real-time simulation system by adding a Simulink subsystem. There are two serious shortcomings with this approach to real-time scripting:

1. Every time a tester wishes to add a test maneuver, the simulation must be stopped; the model must be changed; recompiled and; reloaded onto the real-time simulator.
2. Adding test maneuvers to the model runs the risk of compromising the dynamic portion of the model (ex: a signal line is mistakenly deleted). A disciplined real-time simulation system will have each model validated offline from the real-time simulator and only validated models will be used for real-time testing. Re-validating the model after each maneuver addition quickly becomes very costly.

In order to eliminate the problems discussed above, ADI developed a real-time scripting engine for the ADvantage simulation framework called the "choreographer". Every ADvantage simulation project includes the choreographer functionality unless the user expressly chooses to have it excluded (to improve computational performance for ultra-high-speed simulation). The choreographer is scripted using ADvantageVI. Choreographer scripts are executed in lock
Sequences, Schedules and Triggers

Sequences, schedules and triggers are the main elements of the choreographer. Sequences and schedules are comprised of one or more items. Each item is used to apply stimulus to a particular simulation target variable. Stimulus functions include constant value, the assignment of one variable’s value to the target using the assign item, mathematical functions such as sine wave, and discontinuous functions such as limit. Using the expression item the value of the target variable is controlled by a mathematical or logical expression of other variables in the simulation. Using the playback item, data logged from previous simulations or from field tests may be applied to the target variable.

The key difference between sequences and schedules is in execution. All items within a schedule are executed in parallel. The start and finish time may be user configured. However, all items within a schedule become active when the schedule becomes active.
Sequences, on the other hand, use a wait to control the serial execution of items within a particular sequence. The wait is placed between sequence items and controls item activation based on simulation events or time.

Triggers are the final element included in the choreographer. Triggers are used to start simulation tasks based on simulation time or events. Simulation tasks that can be triggered include schedules and sequences, data logging, and even the start of the simulation clock.

**External Data Dictionary**

Each simulation model in an ADvantage project includes a data dictionary that lists all accessible variables in the model. Often a test requires the use of additional variables not found in the model. These additional variables might be used to count event occurrences, record maximum variable values, be assigned the calculated value of a schedule expression, or many other things. Traditionally these “dummy” variables would be added to the model and used as auxiliary variables for scripting purposes. The two main problems with adding dummy variables to the model are:

1. The need to change a potentially validated model and compromise the validation
2. As the test matrix grows, the number of dummy variables required may also grow and thus require frequent model changes

To solve this “dummy variable” problem, the concept of an external data dictionary was added to ADvantage. External data dictionaries are identical in form and function to the data dictionaries associated with a model. The only difference between traditional data dictionaries and external data dictionaries is that external data dictionaries are independent of any model and can be thought of as a block of real-time variables allocated in simulation memory. Just like model variables, the value of external variables may be queried, changed interactively, logged using the data acquisition features, and scripted using the choreographer.

External data dictionaries add tremendous capability that is consistent with a strict and disciplined test process.
Summary

Real-time simulation has become a mainstay in the development of complex projects. Whether developing a single embedded module or integrating dozens of networked and distributed subsystems, appropriate use of the various categories of test automation will result in highly repeatable and disciplined real-time simulation-based test. This in turn enables development and integration teams to deliver highly advanced electronic systems more efficiently and in a shorter time frame.