An Integrated Test Environment for Systems Engineering

Ryan Brooks
Roberto Escobar

Boeing
Mesa, AZ
What is OSEE’s OTE Framework?

- The **Open System Engineering Test Environment (OTE)** is a framework for requirements-based testing in the context of an overall systems engineering approach.
- OTE has been used commercially to qualify mission software for Boeing’s next generation Apache Attack Helicopter.
- Being contributed as a component of the Open System Engineering Environment (OSEE) Eclipse project.
- Facilitates seamless flow between test development, debugging, execution, and result analysis.
Integrated Requirements-Based Testing

- Bidirectional traceability between software requirements, application code, and tests is provided through OSEE Application Framework
- Coverage holes detection via traceability
- Action Tracking System identifies test impacts driven by requirement changes
- Test development status tracking
- Detailed, integrated test status and coverage reporting
An Integrated Test Environment for Systems Engineering

Architecture

Test Run Manager

OSEE Test Server

I/O Service

Test Environment Service

Models

GPS
Radio 1
Radar
Sensors

Flight Box

1553 Mux Bus

Serial
Ethernet

JINI Lookup

Dynamic Service Discovery

Relational Database
Remote File System

OSEE Application Server

Test Manager
Message Watch
Mux View
Model GUI
Service Manager
Define (Requirements)
ATS (Change Management)

OSEE Test Client

Dynamic Service Discovery

Remote File System

OSEE Datastore

Unit under test containing embedded flight code
Test Environment Service

- Provides dynamic lookup of resources
- Provides both soft real-time and simulated capabilities
- Schedules periodic execution of models (simulation components)
  - API for easy creation of simulated components
- Supports a user configurable number of simultaneous client connections
- Manages I/O and testing resources
Extensible Real-time Messaging System

- Supports communication with real hardware via
  - MIL-STD-1553 MUX
  - Serial
  - Wire
  - Analog and Digital discretes
  - Publish/Subscribe Data Distribution Service (DDS) through Ethernet
Real-time & Simulated Functional Testing

- Simulated Environment (eases demand on limited test station hardware resources)
- Tests (without modification of any kind) can be run in both soft real-time and simulated environments (simultaneously, if desired)
- Simulated components can be used with both environments
- Streaming automated test point tally and rollup of pass/fail determination
- Interactive Testing (automated tests with user input)
Test Manager

- Provides a common interface for functional tests in real-time and simulated environments across all levels of testing fidelity
- Test results streamed in real-time from test service to test manager
- Create, save, and load run lists
- Integrated with JDT/CDT debugger
- Lists available test services that can be used to run test files
- Real-time display of test service usage information
- Supports the execution of multiple simultaneous batches within a single workspace
### Test Manager

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
<th>Status</th>
<th>Output File</th>
<th>Test Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>CND_COM_power.java</td>
<td>PASS (74)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_recv_time.java</td>
<td>FAIL (2/33)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_gps_time.java</td>
<td>PASS (38)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_send_time.java</td>
<td>PASS (35)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_current_time_status.java</td>
<td>PASS (20)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_wod.java</td>
<td>PASS (35)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_mwod_status.java</td>
<td>PASS (35)</td>
<td>READY</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_fmt_trng_net.java</td>
<td>FAIL (6/81)</td>
<td>COMPLETE</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_fmt.java</td>
<td>FAIL (4/65)</td>
<td>COMPLETE</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_conv.java</td>
<td>(0/0)</td>
<td>RUNNING</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_mode.java</td>
<td>IN_QUEUE</td>
<td>IN_QUEUE</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
<tr>
<td>CND_COM_zeroize.java</td>
<td>IN_QUEUE</td>
<td>IN_QUEUE</td>
<td>✓</td>
<td>/iba.test.script.qual.cnd.com...</td>
</tr>
</tbody>
</table>

Selected Host: sun447.com  
Config File Path: C:\Documents\TestManagerEditor.scriptConfig.xml

OFP: Launching...
### Service Manager

- Integrated test server resource management
- Remote viewing and manipulation of test service
Test Output

- Test output file format is XML
- Interactive out files (user can navigate the out file)
- User selectable views of out files
- Automatic correlation of a run-time test point to the test source line that generated it (integrated into JDT using problem view and markers)
- Full test output can be stored in the OSEE object-oriented persistence layer for later analysis and results summary and reporting
### Test Output Editor

**Test Points Linked to Test Code that Generated**

The image shows a screenshot of a test output editor with a table of test results. The table includes the following columns:
- **Title**
- **Expected**
- **Actual**
- **Elapsed Time**

The test cases are listed with a status indicator, and some test cases are marked as failed. The details of the test output are not fully visible, but it appears to be an integrated test environment for systems engineering.

**Filter:**

- **Log:**
  - **LogFile**
  - **Version**
  - **OtcLog**

**Results Summary**


**Test Overview**

- The test output editor provides a summary of test results, including expected and actual outcomes, along with elapsed times for each test case.
Messaging and Playback GUIs

- Monitor, manipulate and record real-time messaging data
- Advanced regular-expression searching for messages
- Import/Export view lists
- Playback environment
  - Supports advanced debugging of the test environment using repeatable conditions
  - Investigate real flight recordings
Questions?