Open System Engineering Environment

The Open System Engineering Environment (OSEE) builds upon the strengths of Eclipse (an open, extensible, platform independent, feature-rich, IDE) to provide a tightly integrated engineering environment that supports lean software and systems engineering. By design, OSEE provides a distributed network-centric engineering environment through its powerful communications framework and distributed services architecture. The OSEE architecture includes a number of specific engineering capabilities including requirements analysis and development, application development, test and simulation and project management. At the heart of the OSEE architecture are the OSEE core services which are available to all the Aspects and enable their tight integration. Since OSEE integrates all engineering areas, the full lifecycle data for a project is managed by a common platform allowing this data to be brought together to form a coherent, accurate view of a project in real-time.

Core Services

- Distributed Computing Framework
- Object Oriented Persistence Layer
- Reporting Framework
- Action Tracking System (ATS)

Requirements Development

- Product Line Engineering
- Author, Navigate and CM Requirements
- Branch, Merge for Multi-Project Support
- Rich Traceability
- System Safety support
- Integrated Data Rights support

Application Development Aspect

- Full-Featured Java, C and C++ IDE
- Automated Unit-Testing Framework
- Full and Integrated CM

Test and Simulation Aspect

- Java and C++ Based Test Scripts
- Automated Real-Time Functional Test Framework
- Advanced Test Manager and Environment Service
- Monitor, manipulate and record real-time messaging data

Realized Objectives

Single Environment
Tightly Integrated Tools
Common Look and Feel
Reduced Tool Development Cost
Reduced Lifecycle Costs
Reduced Multi-Customer Costs
Easier Bi-Directional Traceability
Platform Independent
Different Views for Each Role
Open Standards
Open and Extensible Platform
Zero cost licensing model
Integrated CM
Rich Core Service Layer
Collaborative Boeing Solution

Project Management Aspect

- Configuration Management
- Metrics
- Charting and Graphing
- Integrated Help and Training

Ryan D. Brooks   Donald G. Dunne
ryan.d.brooks@boeing.com   donald.g.dunne@boeing.com

OSEE Test and Simulation

Fully automated testing

- Automated test point tally and rollup of pass/fail determination
- Unit and Functional Testing
- Java and C++ Based Test Scripts
- Automated Display Testing (ADT)
- Simulated Environment (eases demand on limited test station resources)
- Tests (without modification of any kind) can be run in both real-time and simulated environments (simultaneously, if desired)
- Real-time messaging system supporting Mux, serial, analog and digital discretes, and publish/subscribe
- Interactive Testing (automated tests with user input)

Output files

- Output file in XML
- Interactive HTML outfile viewer (user can navigate the out file) with normal, debug, and flat views
- Direct linking of output file lines to source test lines
- Automatic correlation of a run-time test point to the test source line that generated it
- Built-in programmatic bidirectional traceability between requirements, application code, and test
- XML output file can be automatically transformed into any format
- Can apply any XSLT translation

Test Source File

- Built in traceability to software requirements
- Test files are color/tool tip annotated with failures

Tightly Integrated into the Open System Engineering Environment

- Leverages Java Development Toolkit (JDT) and C/C++ Development Toolkit (CDT)
- Facilitates seamless flow between test development, debugging, execution, and result analysis
- Supports the execution of multiple simultaneous batches within a single workspace
- Built-in help system extended with test manger user guide

Test Manager GUI

- Consistent, clean interface for the control of unit and functional test in the real-time and simulated environments
- Test results streamed in real-time from test service to test manager in Single or Batch run model
- Remote viewing and manipulation of test service
- Enable/disable running of individual scripts in the run list
- Drag and Drop test files into run list
- Integrated with code debugger
- Access to test and output files from the GUI
- Lists available test services and their usages that can be used to run test files
- Jini based communication

Message GUI

- Monitor, manipulate and record real-time messaging data
- Advanced regular-expression searching
- Import/Export view lists
- Remote attachment to message service

Test Environment Service

- Utilizes Jini – Java's distributed computing framework
- Provides dynamic lookup of resources
- Provides both real-time and simulated capabilities
- Schedules periodic execution of models (simulation components)
- Supports user configurable number of simultaneous connections

Ryan D. Brooks **Donald G. Dunne**
ryan.d.brooks@boeing.com donald.g.dunne@boeing.com