Workshop Papers

2003 European Simulation Interoperability Workshop
(EURO-SIW)

Sponsored by the Simulation Interoperability Standards Organization (SISO)

www.sisostds.org
Verification and Validation - Process Definition and Tool Development Within Euclid RTP 11.13 (03E-SIW-063)

Andreas Nyberg
SaabTech Systems AB
Nettovagen 6
175 88 Jorfalla
+46 8 580 843 24
annya@systems.saab.se

Richard Harris
Thales Training & Simulation
Garwick Road, Crawley
West Sussex, RH10 9RL
+44 1293 563806
richard.harris@thales-tts.com

Wim Huiskamp
TNO Physics and Electronics Laboratory
P.O. Box 96864
The Hague, 2509 JG
+31 70 3740274
huiskamp@fel.tno.nl

João Pereira
INESC-ID
Rua Alves Redol, 9
Lisbon, 1000-029
+351 21 3100218
jap@inesc.pt

Keywords:
Verification, Validation, Euclid 11.13, SEDEP, FEDEP

ABSTRACT: Euclid RTP 11.13 is a major European initiative to promote the use of Synthetic Environments (SEs). The aim of the project is to "overcome the obstacles that prevent SEs being exploited in Europe by developing a process and an integrated set of prototype tools that will reduce the cost and timescale of specifying, creating and utilising SEs for collective training, mission rehearsal and simulation based acquisition". The SE Development & Exploitation Process (SEDEP) is the EUCLID RTP 11.13 process that has been developed for creating and utilising SEs. The SEDEP is based on FEDEP, but extends it by providing a better interface with the user and enhances it with respect to improved support for iterative federation development and re-use of existing components. The Repository lies at the heart of the SEDEP and provides the facility for storing and sharing data about a SE project.

Verification and Validation (V&V) is a vital part of any development program and has been addressed as a significant issue during the development of the SEDEP. Within the SEDEP, Validation is defined as the process that ensures that a (final or intermediate) product meets the user requirements, while the Verification process ensures that it meets the system requirements. Although V&V (especially Validation) takes place during most of the development process, the main part of V&V in SEDEP, lies in the step "Integrate & Test Federation": The SEDEP process describes that the purpose of this step is "to initialise and configure the identified federates, establish all required interconnectivity between federates, and test federates and the federation prior to execution".

434
This paper describes Euclid 11.13's views regarding Verification and Validation, the obstacles identified in current V&V processes and the solutions to overcome them. This includes both the entire V&V process, from defining the V&V plans to the final system validation, and the Euclid Federate Test Suite, which aids the user in performing the various activities in the V&V process. The Federate Test Suite guides the user through the process of V&V as well as performing regular functional testing. The Euclid Federate Test Suite provides strong support for automated (scripted) testing and an iterative approach to the testing process. It consists of both in-house developed components and COTS tools, which are all interfaced to the common Repository.

1. Introduction

Verification and Validation (V&V) is vital to any program, large or small. There is not much that is worse than putting a lot of resources and heart into a program and having its result crash upon first contact with the customer. V&V is the process that makes sure that products work as the customer intended. V&V has however, more often than not, been approached with a sigh rather than a smile. The process of going through V&V has usually been seen as a cumbersome necessity rather than a chance to make sure the final program outcome will be as good as possible.

The problems of V&V and possibilities for improvement of this process have been addressed by Euclid RTP 11.13. This program is a major European initiative to promote the use of Synthetic Environments (SEs). This is done by the development of two major products. The first product is the SE Development & Exploitation Process (SEDEP), a process for creating and utilising SEs. The SEDEP is based on FEDEP, but extends it by providing a better interface with the user and enhances it with respect to improved support for iterative federation development and re-use of existing components. SEDEP consists of eight steps. These are shown in Figure 1.

![Image of SEDEP steps and products]

Figure 1. SEDEP Steps and Products

V&V belongs to Step 5, Integrate and Test Federation. This step is responsible for integrating the Federation elements with the RTI and testing the Federation ready for operation. This includes testing the interactions between Federates, verifying the Components against the system requirements, and validating the Federation against the user requirements.

Where appropriate, the SEDEP can be iteratively applied to support an SE project. The depth to which the process is applied during each iteration is dependent on the quality of information provided. An important feature of the SEDEP is that all the information is stored in the Repository, so that successive iterations of the process can build on the information already available. The recent update of the
IEEE FEDEP has included concepts developed within the SEDEP.

The second Euclid 11.13 product is the SE development environment. It consists of a large set of tools, all aiding the process of SE development. At the centre of the environment lies the Repository, which ties all the tools and processes together. It is the combination of all the different components that enables the facilitation of producing and reusing future and past SEs. For more information about Euclid RTP 11.13, see [1] and [2].

2. The Euclid 11.13 Verification & Validation Process

For Euclid 11.13, verification and validation is based on the testing process put forward by the Georgia Institute of Technology in [3] and [4]. This process has five steps, starting with the testing of individual federates, moving through integration to the testing of the entire federation. Figure 2 illustrates the sequence of these steps.

![Federation Test Process Diagram]

Figure 2. Federation Test Process

There is some debate in the literature over quite where the boundary falls between federate testing and federation testing. For the purposes of Euclid 11.13, this boundary is shown by the dotted line in Figure 2. Conformance and Application Testing are considered to test individual federates while the remaining steps test part or all of a federation. The five process steps may therefore be summarised as follows.

- **Conformance Testing** – Testing that a single federate communicates using the correct HLA “syntax”, ie, using the HLA interface Specification, Object Model Template (OMT) and HLA Rules.

- **Application Testing** – Testing that a single federate communicates using the correct, specified federate semantics, ie, that data being communicated is meaningful and appropriate to the federate.

- **Integration Testing** – Testing that two or more federates communicate correctly within a federation, according to both HLA “syntax” and federation semantics, ie, that data being communicated is meaningful and appropriate to all federates concerned.

- **Functional Testing** – Testing that all federates in a federation interact as required by the federation, ie, that they are able to affect and be affected by one another in the ways required by the federation.

- **Scenario Testing** – Testing that all federates in a federation interact as required by specific exercises, ie, that they are able to affect and be affected by one another in the ways required by specific exercises.

More details about this testing process, along with examples for each step, can be found in [3].

This testing process can be considered to cover both verification and validation. For Euclid, verification is defined to mean “testing against system requirements”, while validation is defined to mean “testing against user requirements”. It was originally assumed, therefore, that there was a clean boundary between verification and validation. This simplistic split between verification and validation is shown by the dotted line in Figure 3. However, as Figure 3 also shows, further analysis revealed that there is actually a significant overlap between verification and validation, with several of the test steps including both types of testing.

![Verification & Validation Processes Diagram]

Figure 3. Verification & Validation Processes

Consequently it was seen to be important to consider verification and validation together, applying allied processes and tools to both in a coordinated manner. Euclid 11.13 therefore provides coordinated sets of
tools for both verification and validation, as shown in Figure 4.

Figure 4. Verification & Validation Test Suites

3. Tools Selection For Euclid 11.13

Testing carried out under the Euclid 11.13 verification and validation process is based on a simple basic approach. Federate(s) Under Test (FUTs) are stimulated, using RTI service calls via the HLA, and the subsequent HLA traffic and federate behaviour is analysed to determine if the federate(s) have responded in the required manner. The types of stimulation applied, and the expected responses, are directed by the testing step being executed, the HLA capabilities of the federate(s) in question and the requirements against which the testing is being carried out.

The Euclid verification and validation tools therefore need to provide three main capabilities to support this approach:

- Stimulation of the Federate(s) Under Test (FUT) via the HLA
- Analysis of HLA traffic
- Test Session Management – coordinating the test sessions, directing the tests to be run and accumulating the results.

A number of tools and suites of tools already exist that claim to provide these capabilities. Such tools were therefore considered for use before deciding on what additional tools development could beneficially be carried out by the Euclid programme.

3.1. Available Off The Shelf Tools

The most significant off the shelf tools identified by Euclid 11.13 are the suites of testing tools shown in Table 1. Unfortunately these have all been generated for use exclusively by the US government for their own accreditation programmes and proved to be unavailable outside the agencies concerned.

<table>
<thead>
<tr>
<th>Tool(s)</th>
<th>Source</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federation Test System (FTS)</td>
<td>GTRI/DMSO</td>
<td>[3]</td>
</tr>
<tr>
<td></td>
<td>Acusoft/STJCOM</td>
<td>[5]</td>
</tr>
<tr>
<td>Federate Compliance Testing Tool (FCTT)</td>
<td>GTRI/DMSO</td>
<td>[6]</td>
</tr>
<tr>
<td>RTI Verifier</td>
<td>MITRE/DMSO</td>
<td>[7]</td>
</tr>
</tbody>
</table>

Table 1. Existing Comprehensive Test Tools

Other tools exist that provide one or other of the desired capabilities. These are shown in Table 2. These tools are available "off the shelf", either Commercially (COTS) or from the US Government (GOTS).

<table>
<thead>
<tr>
<th>Tool</th>
<th>Source</th>
<th>Capability</th>
<th>Ref</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Federate</td>
<td>DMSO</td>
<td>Stimulator</td>
<td>-</td>
</tr>
<tr>
<td>Federation Verification Tool (FVT)</td>
<td>DMSO</td>
<td>Analyser</td>
<td>[8]</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[9]</td>
</tr>
<tr>
<td>Object Model Development Tool (OMDT)</td>
<td>DMSO</td>
<td>Analyser</td>
<td>[10]</td>
</tr>
<tr>
<td>Visual OMT</td>
<td>Pitch</td>
<td>Analyser</td>
<td>-</td>
</tr>
<tr>
<td>Test Executive</td>
<td>Pitch</td>
<td>Stimulator</td>
<td>-</td>
</tr>
<tr>
<td>Test Generation and Tracking Tool</td>
<td>Aegis</td>
<td>Analyser</td>
<td>[11]</td>
</tr>
<tr>
<td>Test Recording Tool (TRT)</td>
<td>TT&amp;S</td>
<td>Manager</td>
<td>-</td>
</tr>
</tbody>
</table>

Table 2. Off The Shelf Test Tools

3.2. Selected Tools

Since a selection of these off the shelf tools could be used to provide the core capabilities of the Euclid verification and validation tools, it was decided to base the Euclid tools suite on the most appropriate of these COTS and GOTS tools. The tools selected are shown in Table 3. Together with additional Euclid-developed tools described below, these tools form the Federate Test Suite (FTS), as shown in Figure 5.

1 The RTI Verifier is considered here since the tests required to verify an RTI are essentially the same as those required to verify a federate; the problem being addressed in each case is essentially the same, just viewed from opposite ends.
The use of these COTS/GOTS tools has three advantages for Euclid. It allows Euclid to leverage work done by existing players in the HLA community, rather than reinvent already extant wheels. This in turn allows the Euclid programme to concentrate on adding value to these currently available facilities. Finally, the existing tools provided an initial tools suite that could immediately be used for process development work, prior to Euclid-specific enhancements. Together these factors will serve to maximise the return on Euclid investment.

### 4. Tools Development for Euclid 11.13

The following additional tools developments are being made by the Euclid 11.13 programme. These tools are intended to complement the off the shelf tools and streamline the application of the Euclid verification and validation process.

---

2 The Federation Execution Planner’s Workbook (FEPW) is used by the FVT in its analysis. The FEPW Editor is therefore included as part of the tools suite as it provides the capability to verify the contents of the FEPW prior to testing of the FUT.
4.1. Verification Analysis Tool (VAT)

The off-the-shelf Federation Verification Tool (FVT) is used by the FTS to analyze network traffic. However, while the FVT provides a helpful overall summary of HLA activity and any discrepancies therein, such results still require analysis by a human operator to determine their overall significance. Euclid 11.13 is therefore developing a tool to further analyze the FVT output and simplify or remove the need for operator involvement. This tool is called the Verification Analysis Tool (VAT).

While the Federate Under Test (FUT) is stimulated using the Test Federate (TF) tool, the FVT monitors the FUT's update/reflect and send/receive activity and the RTI service calls that are used. As well as displaying this data to the operator, the FVT can also write it to two text files. The VAT reads the two FVT-generated text files, analyses their contents and determines whether the FUT has passed the tests. It reports the results to the operator and may also write these results to an XML file.

4.2. Test Navigation Tool (TNT)

The Test Navigation Tool (TNT) is the test session manager of the Federate Test Suite (FTS) and assists the operator in composing, managing and tracking the tests.

The TNT is initially used to author the tests that are to be carried out. The relevant user and system requirements are loaded into the TNT and tests may then be composed for each of these requirements. Where previous tests can be reused, these may also be loaded and updated or used as they are.

During testing the TNT leads the operator through the tests step by step. Note that the tests are not run in an automatic fashion. This makes the TNT very flexible, allowing it to be applied to tests that would not easily lend themselves to an automated approach. However, where tests can be run in an automated fashion, automation is provided by the other tools in the FTS, principally the Test Federate. Where possible the operator will interact with the TNT during testing, but if necessary, printed copies of the tests may be generated for portability around large SE developments. These may also be used delivery to deliver results to subject experts where applicable.

Once each test is complete, the operator can record the results via the TNT interface. For a test to be passed, the operator must successfully complete all the actions dictated by each test. The TNT shall indicate the status of testing with an appropriate pass/fail grading. Finally the database of tests and results may be saved.

4.3. Interface To The Euclid Repository

Euclid 11.13 defines a Repository containing all data related to federations being developed. The TNT will be connected to this Repository in order to retrieve the user and system requirements that are to be tested as well as to store, retrieve and create tests and their results, see figure 6. A JAVA based interface provides access from the TNT to the Euclid Repository. A DLL provides the required services to fulfill the access in both directions.

*Figure 6. TNT Interface to the Repository*
SOAP is used to access the Repository. All the information exchanged with the Repository is in XML format so DTDs were developed to define new entities within the Repository. When the TNT needs to exchange information with the Repository, it calls a function or functions within the communication interface that trigger read and write operations either on the TNT side or on the Repository side. The interface module translates messages from TNT (Access) format to Repository format and vice-versa. It also establishes a transparent communication media for the TNT: the TNT asks to establish a connection and the module handles all communication issues after that.

4.4. Verification & Validation Process Guide

Existing literature on actually carrying out verification and validation of federates and federations typically falls into four categories:

- References against which to test, eg [12], [13], [14]
- Descriptions of testing processes, eg [3], [4]
- Descriptions of testing tools, eg [5], [6], [9], [10], [11]
- Directions in getting federates certified by a third party, eg [6], [15], [16]

There appears to be little in the way of step by step guides on how to use the available tools to set up and execute specific tests according to identified testing processes. In addition, of course, no material exists to advise on the application of the tools being developed by Euclid 11.13.

A Verification & Validation Process Guide is therefore being written for Euclid 11.13. This will provide an operator with a practical step by step guide to carrying out verification & validation by applying the Euclid V&V process using the Federate Test Suite (FTS). It will draw on information provided by the existing literature and will complement individual tool user guides by directing the operator in when and how specific tool capabilities should be used.

5. Testing for Euclid 11.13

Verification and Validation for Euclid 11.13 will use the Federate Test Suite (FTS), applying it according to the Euclid 11.13 V&V process, as directed by the Euclid 11.13 V&V Process Guide.

Firstly, the tests that are to be run must be written, based on the user and system requirements. This is done using the Test Navigation Tool (TNT). The requirements are extracted from the Euclid Repository by the TNT, which displays them to the operator. Suitable tests may already exist within the Repository, in which case they can be extracted and reused. Otherwise, the necessary tests may be defined in the TNT against each requirement. These tests are broken down into test cases and detailed test procedures and are stored in the TNT database in relation to the requirements that they address. Where predefined sequences of RTI service calls form part of these tests, these may be scripted using the Test Federate (TF). Finally the tests, along with any supporting Test Scripts from the TF, are stored in the Repository for later use.

Testing will be driven in all cases by the TNT, acting as the test session manager. Once the tests have been loaded from the Repository, the operator will select the tests to be run and be led step by step through each test by the TNT.

The TNT will initially prompt the operator to set up the test. eg to start up all federates and test tools required for the test. This will include testing the FEPW file required by the Federation Verification Tool (FVT), the FOM and the SOM belonging to the Federate(s) Under Test (FUT). The FEPW file is validated using the FEPW Editor and the FOM and SOM(s) using the OMDT tool.

The Test Federate (TF) will then be used to stimulate the FUT. This will involve the operator either specifying individual RTI service calls or, more often, running the pre-defined Test Scripts using the TF.

Some tests may require the operator to observe the results of particular stimulations. This may involve, for example, observing the FUT’s human interface if it has one; observing the FUTs appearance on a 3D Stealth or 2D Map display; or observing the GUI on the FVT. The TNT will indicate any observations that the operator should make, along with the results he should find. Once each test case has been run, the TNT will request the operator to enter the results.

Once a test session has been completed, the FVT and Verification Analysis Tool (VAT) will be used to analyse the HLA traffic recorded during the testing. This will involve the Operator being directed to write out a Summary file and Discrepancy file from the FVT. These files will be reviewed automatically by the VAT and the operator will enter the overall VAT-generated result into the TNT.
The results entered into the TNT will be those observed by the operator, mainly via the GUIs of the various tools. These will be included in a Test Report generated by the TNT. Finally, the test results will be stored in the Euclid Repository. From there they may be retrieved at a later date for activities such as updating the FUT(s) to address failed tests or proving the FUT(s) have successfully passed their tests. They may also be used as starting point for later testing if the FUT(s) are modified in the future.


The scope of the Euclid 11.13 programme is to build mainly prototype based tools that would highlight the process and its ideas, and not to build full fledged complete tools. The components in the V&V Tools Suite have therefore been planned and developed in phases. The first phase will be the release phase of the tools, accompanying the completion of the Euclid 11.13 programme in Fall 2003. The different V&V tools will have the basic functionality to enable a developer to put his/her component through the Euclid 11.13 V&V process.

For the second phase, the increased functionality needed for making the tools more useful has been described and planned. However, it is expected that full development of the tool suite will fall outside the scope of Euclid11.13.

7. Conclusion

The Verification and Validation process has received a lot of attention by Euclid 11.13 during the development of the SEDEP and its accompanying tool environment. A main focus for the developing of the V&V process and accompanying tools has been to reduce the work load for the people developing programs in general and those participating in the V&V process in particular. This has been done by developing a tools suite containing both COTS and in-house developed components and presenting the V&V process in a user-oriented manner. The result is a tools suite that functions as both a V&V guide and manager as well as helping the test engineers to prepare and conduct the actual tests.

8. References


Author Biographies

ANDREAS NYBERG is a Systems Engineer at the Modeling & Simulation Department at SaabTech Systems. For the last five years, he has been working within the distributed simulation field. Specializing on HLA from both an architectural and an implemental view. He is currently involved in Euclid 11.13 as well as working within the V&V field of the development of a C2-system for a Helicopter Platform.

RICHARD HARRIS graduated from Imperial College, London, in 1987 with a BSc (Eng) honours degree in Computing Science. He has worked in the simulation industry for 15 years, with Link-Miles and Thales Training & Simulation (TT&S). He is currently the technical lead in TT&S’ Multi-Sensor Database Tools group, where he combines development of database modelling tools with work on SEDRIS and Euclid 11.13.

WIM HUISKAMP is a research scientist in the Command & Control and Simulation Division at TNO-FEL. He holds a M.Sc. degree in Electrical Engineering from Twente University of Technology, The Netherlands. He works in the field of High Performance Computing and Networking and specialises in design and implementation of distributed computer architectures. His research interests include system architecture, real-time simulation and autonomous vehicle technology. Wim was the project lead for several national and international simulation (and simulation interoperability) projects.