Acceptance Testing Procedure for Network Enabled Training Simulators

Peter W. Ross
Peter D. Clark
Air Operations Division
Defence Science & Technology Organisation (DSTO)
PO Box 4331, Melbourne, Victoria, 3001, Australia
+61 3 9626 7615
peter.ross@dsto.defence.gov.au

Keywords:
Distributed Simulation, Simulators, Software Engineering, Training, Testing

ABSTRACT: *Acceptance testing is a necessary stage in any complex procurement, as it determines whether the supplier has satisfied the requirements of the contract. Over the next ten years the Australian Department of Defence will acquire many new platform training simulators that will support distributed team training, otherwise known as network-enabled training simulators. This form of training is made possible through the use of distributed simulation standards. It is necessary to ensure that new simulators comply with the relevant distributed simulation standards during acceptance testing. However, at present there is no uniform procedure for acceptance testing of these network-enabled simulators. This paper introduces distributed simulation concepts in relation to platform training simulators and presents an acceptance testing procedure that is based on the authors' prior experience with testing of training simulators. The procedure will facilitate acceptance and interoperability testing conducted on behalf of the Australian Defence Material Organisation by the Defence Science and Technology Organisation (DSTO). Present activities include testing of the Royal Australian Air Force's AP-3C Advanced Flight Simulator and the Royal Australian Navy's FFG Upgrade Team Trainer, FFG Upgrade On Board Training System; and Super Seasprite simulator.*

1. Introduction

Over the next ten years the Australian Department of Defence will acquire several new platform training simulators that support distributed team training. For distributed team training to be reliable and cost effective, and therefore embraced by the user, simulators must be network interoperable. The risk of non-interoperability is reduced by thoroughly testing simulators against the relevant distributed simulation standards.

A majority of the new Australian platform training simulators will support the Distributed Interactive Simulation (DIS) standard. These include the AP-3C Advanced Flight Simulator, Airborne Early Warning & Control (AEW&C) Operational Mission Simulator, Armed Reconnaissance Helicopter (ARH) simulator, Super Seasprite simulator, and FFG Upgrade Project Onboard Training System (OBTS). Several simulators supporting the High Level Architecture (HLA) standard will be delivered in the future, including the F/A-18 Hornet Aircrew Training System.

Whilst existing network-enabled training simulators, including the Royal Australian Air Force (RAAF) AP-3C Operational Mission Simulator, Air Defence Ground Environment Simulator, and Royal Australian Navy (RAN) FFG and ANZAC operations room team trainers, have supported the DIS standard, the requirements specification and acceptance testing procedures have varied. Consequently the capabilities and limitations vary greatly among simulators, due often to inadequate requirements specification, varying model resolution, and defects present in the delivered product.

To reduce this risk for new acquisitions, Air Operations Division (AOD), DSTO, has undertaken research to identify minimum interoperability requirements and issues relating to the implementation of network-enabled training simulators [1],[2]. Similar efforts have been undertaken by other nations [3]. Prior to the present DSTO initiative, there was no uniform procedure within the Australian Department of Defence for this form of testing.

2. Distributed Team Training

The term 'platform training simulator' is employed by AOD to describe a human-in-the loop training simulator that models the virtual battlespace at the tactical level in real-time. Whilst there are no set rules for simulator design, a generic simulator normally consists of five components, that are physically dispersed throughout the training facility, namely;

Trainer. The component/s manned by the trainee/s, for example operating consoles, cockpit, operations room, or bridge. The platform, which the trainer represents, is referred to as the ownship, or the "ownship entity" within the simulation.

Control station(s). The component/s used to configure the simulator and control execution of a training exercise. Standard functions include defining the reference point (or game centre), starting and stopping the exercise, and manually repositioning the ownship.

Instructor/Asset station(s). The component/s that manages additional entities within the exercise, such as those representing the red force. Traditionally these stations have been manned by instructors and the additional entities controlled using low-level, semi-automated behaviours. There is a move, however, to reduce manning requirements through the use of intelligent agent technology. The instructor station may also incorporate functionality of the control station or debrief components.

Debrief. The component that provides performance feedback to the trainee/s following the execution of an exercise.

Simulation Computer(s). The component/s that perform platform dynamics, sensor and emitter modelling, and display rendering calculations.

2.2 Distributed Simulation

In this context, distributed simulation is the provision of a shared virtual battlespace, in which trainees interact in order to achieve training objectives. Information representing the virtual battlespace is known as "ground truth" and is exchanged over a data communications network. This information is perceived independently by each simulator.

The way in which a simulator internally models the virtual battlespace is called the *internal model*. The internal model is often different for each training simulator, for example one simulator may consider the earth's surface to be flat, whilst another may model it as an ellipsoid. The internal model is a direct result of the simulator's functional requirements and corresponding engineering design decisions. To conduct distributed team training, a standard model is required for all participating simulators. Rather than forcing all simulators to behave in the same manner, a secondary model, known as the *network model*, is used.

Simulation models, be they internal or network, are composed of objects and/or interactions. An object describes information that is persistent for some duration of the simulation, whereas an interaction describes an instantaneous event. Though these network model concepts are present in most distributed simulation standards, the terminology often varies.

The network model is purely a conceptual representation of the virtual battlespace, and does not define how objects and interactions are exchanged between simulators. The exchange process is instead defined by the *network protocol*, also known as the messaging or wire protocol. The network protocol often leverages existing network transport technologies, such as Internet Protocol (IP) or Asynchronous Transfer Mode (ATM).

Established distributed simulation standards, including SIMulator NETworking (SIMNET), Distributed Interactive Simulation (DIS) and the Aggregate Level Simulation Protocol (ALSP) define a baseline network model and protocol. More recent standards, including HLA and the Test and training ENabling Architecture (TENA), leave the definition of the network model and protocol open as an engineering design decision. These design decisions, if not appreciated, lead to non-interoperability. To counter this, effort has been made to establish reference or base-line network models, that whilst not ensuring interoperability, encourage reuse and commonality between simulations [4].

2.3 Distributed Simulation Interface

Network enabled training simulators incorporate a sixth component, in addition to the generic simulator components identified above. This distributed simulation interface component performs two tasks. The first is translation, where information represented by the internal model is translated into a network model representation, and vice-versa. Information is often discarded, augmented or converted during the translation process; coordinate system conversion, for

example, is almost always required. The second task is exchange, where information represented by the network model is marshalled and sent to other hosts using the network protocol, and conversely received and un-marshalled. Table 1 compares the conceptual layers of a generic distributed simulation interface for DIS and HLA against the International Standards Organisation Open Systems Interconnection (ISO/OSI) network model [5].

Table 1: Conceptual layers and tasks of a distributed simulation interface

<table>
<tr><th>Layer</th><th>DIS</th><th>HLA</th><th>ISO/OSI</th></tr>
<tr><td>Internal model</td><td>Internal model</td><td>Simulation Object Model</td><td>Application</td></tr>
<tr><td>↕</td><td colspan="2">Translation</td><td>↕</td></tr>
<tr><td>Network model</td><td>PDU types</td><td>Federation Object Model</td><td>Application</td></tr>
<tr><td>↕</td><td colspan="2">Exchange</td><td>↕</td></tr>
<tr><td rowspan="2">Network protocol</td><td rowspan="2">Byte order, Structures, Heartbeats, Timeouts</td><td rowspan="2">Run Time Infrastructure</td><td>Presentation</td></tr>
<tr><td>Session</td></tr>
<tr><td rowspan="4">Network transport</td><td rowspan="4">User Datagram Protocol / IP</td><td rowspan="4">Typically IP</td><td>Transport</td></tr>
<tr><td>Network</td></tr>
<tr><td>Data Link</td></tr>
<tr><td>Physical</td></tr>
</table>

Objects and interactions generated by the simulator flow down through the layers, whereas objects and interactions generated by remote simulators flow up through the layers. The former is referred to as sending, and the latter as receiving. When the distributed simulation interface is not used, the simulator is said to be operating in stand-alone mode.

2.4 Interoperability

Interoperability is defined as the ability of two or more systems or components to exchange information, and to make appropriate use of that information [6]. During the development of DIS and HLA, networked simulator interoperability was decomposed into three distinct levels: compliant, interoperable and compatible [7],[8].

Compliant. A simulator is considered to be compliant if the distributed simulation interface is implemented in accordance with the relevant standards. This is achieved at the acceptance testing stage, by ensuring that the translation and exchange tasks are performed correctly.

Interoperable. Two or more simulators are considered to be interoperable if they can participate in a distributed training exercise. This is achieved at the requirements specification stage, by ensuring that each simulator is built to equivalent network model and protocol standards. Engineering design decisions relating to the choice of network model and protocol should be reviewed thoroughly, as these directly influence this level of interoperability.

Compatible. Two or more simulators are considered to be compatible if they can participate in a distributed training exercise and achieve training objectives. This is achieved at the training needs analysis stage by ensuring that the capabilities and performance of each simulator are sufficient to meet training objectives. The expression “fair fight” is frequently used to describe compatibility.

These definitions demonstrate that a compliant simulator will not necessarily be interoperable with other compliant simulators, and likewise, that just because two or more simulators are interoperable, they are not necessarily compatible for training.

3. Solution - Acceptance Testing Procedure

The objective of acceptance testing is to establish that the supplier has satisfied the requirements of the contract, therefore mitigating the risk of defects or other inadequacies throughout the project’s operational lifetime. It occurs prior to ownership of the project deliverable being handed over to the customer (the Commonwealth of Australia), and is conducted in the intended operational environment (the training facility), as opposed to the supplier’s development environment. Ideally, few defects should be identified at the time of acceptance as modern software engineering practices encourage testing throughout the product development cycle [9]. Unfortunately such practices are not always adopted, or if adopted, are awarded lower priority in the rush to meet delivery schedules.

Thorough testing of a simulator’s distributed simulation interface is required for three reasons. Firstly, distributed simulation protocols are often intolerant to implementation faults; one incorrectly set field (or data bit within a field) maybe sufficient to prevent distributed team training, or lessen its effectiveness. Secondly, distributed simulation standards are often ambiguous and incomplete to some degree, meaning that two standards compliant simulators may be non-interoperable due to the suppliers forming different interpretations of the

standard's intent. Finally, the defects are seldom apparent until the distributed simulation interface is used in anger. The cost of resolving defects at short notice for an exercise is often prohibitive and training quality generally suffers as a consequence.

Contract requirements often specify implementation to a subset of distributed simulation standards, as opposed to interoperability with a specific simulator. For this reason, as alluded to in section 2.4, acceptance testing can only guarantee compliance. Interoperability and compatibility is achieved through consistent requirements specification, although a uniform testing procedure serves to reduce the risk of non-interoperability.

The time and resources allocated to acceptance testing are often limited; therefore the procedure needs to be comprehensive, efficient and repeatable. The procedure employed by DSTO consists of three stages and is detailed in the following sections.

3.1 Planning

Planning identifies the aspects of the simulator to be tested, level of manning required to operate the trainer and/or instructor stations, and the anticipated duration of testing. Often a simple approach is taken, where testing of all functionality related to the distributed simulation interface is proposed. As in the planning for a distributed training exercise, agreement must be reached on data, including platform types and the location within the virtual battlespace whereby testing will take place. Deployment and set-up of the test equipment, including data classification and network media compatibility, must be also considered.

Given that the distributed simulation interface shares connectivity with other components of the simulator, it is desirable to perform distributed simulation tests following preliminary acceptance of the stand-alone simulator. Otherwise, the results of testing may be influenced by defects present in the stand-alone simulator.

3.2 Test Activity

The test activity occurs at the training facility and often spans several days, depending on the amount of testing proposed in the planning stage. The black box testing methodology, which evaluates the functionality or performance of the system irrespective of internal implementation details, is employed. Figure 1 shows the black box view of a generic training simulator, where the exposed interfaces are the Human Machine Interface (HMI) and Network Interface Card (NIC).

The functional requirements are tested by stimulating the black box with input actions and witnessing the resulting outputs. This is performed in an iterative manner using a library of test cases tailored to the distributed simulation standards supported by the simulator. Test cases are categorised into three types:

- Configuration testing verifies that the simulator can be configured appropriately for a distributed training exercise.
- Send testing verifies that information sent by the simulator complies with the relevant simulation standards. The input actions for send tests normally relate to the HMI.
- Receive testing verifies that the simulator responds correctly to information generated by remote simulators. The input actions for receive tests normally relate to the NIC or network model.

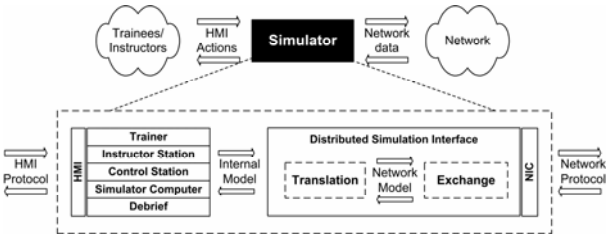

Figure 1: *Black box view of a training simulator*

It is desirable to perform testing in the order listed above as this enables an understanding of the simulator's capabilities to be acquired through a passive analysis of the network data, prior to sending information to the simulator.

Certain test cases, such as dead reckoning accuracy tests, require detailed analysis of the witnessed output, and are best performed following the test activity (for example, in a laboratory environment) to make more efficient use of time with the simulator. To facilitate this, relevant HMI actions and network data sent and received by the NIC are recorded in a test log, which is a combination of written notes and data files, where log entries are time stamped to enable correlation of events.

3.3 Documentation

Following the test activity, a document is produced that details the results of testing. The report can be styled as either a formal report, that introduces the simulator and

describes the outcomes of test activity, or a compilation of individual incident reports, where each cites the outcome of a specific test case.

Regardless of the style used, each problem identified is highlighted by severity, and the potential impact on training effectiveness explored in terms meaningful to the project authority. DSTO currently employs a three tier severity rating scheme, where a FAULT indicates non-compliance that prevents interoperability with another simulator, and resolution is advised. An ISSUE indicates non-compliance, however the problem is unlikely to prevent interoperability, and therefore resolution is desirable. An ACTION indicates the need for further testing as the severity of the problem is unknown, for example, due to contradictory test results.

Ultimately the report indicates whether the project authority should accept the distributed simulation component of the simulator, and if not, makes recommendations for change. If significant problems are identified, the relevant test cases should be repeated to ensure that the supplier makes appropriate corrections.

4. Test Case Development

Test cases serve to demonstrate the implementation of individual distributed simulation requirements. There are several types of requirements for distributed simulation, as shown in Table 2. An example network model requirement may stipulate "simulation of Identification Friend or Foe (IFF) transponder mode 3". Each requirement type differs in terms of complexity, test case development methodology and the equipment suitable to facilitate test execution.

Given that distributed simulation standards are often ambiguous, it is necessary for the test engineers to have a clear and consistent understanding of the standards requirements. In related research, DSTO has documented known ambiguity and established interpretations of the DIS standard, and is actively involved in the development of a revised standard [10],[11]. To add authority and assist defect resolution, test cases should reference the original requirements text, and state any interpretations applied.

Table 2: Distributed simulation requirements

Requirement	**Suitable test equipment**
Network hardware	Another network device
Network transport	Transport manipulation utilities
Network protocol	Object and interaction generation
Network model	and instrumentation equipment
Training	Scenario generator, or another training simulator

Network transport and hardware requirements are normally evaluated using a small number of test cases, for example, to demonstrate Internet Control Message Protocol (ICMP) ping replies, network address and port configuration, and hardware compatibility with other network devices, such as switches, hubs and routers.

For each network protocol requirement, test cases are developed to demonstrate exchange of data, for example, packet heartbeat intervals, byte ordering and data structure placement. Because the network protocol is frequently synonymous with the network model, these tests are carried out in parallel with network model tests. However, for some distributed simulation standards, it is possible to independently test the network protocol implementation [12].

For each network model requirement, the related objects and interactions are identified, and test cases written for relevant field permutations, with respect to send and receive testing. This is done to exercise all relevant software execution paths. For example, the IFF requirement above would be evaluated with at least four test cases, in order to demonstrate sending and receiving of mode 3 when the transponder is enabled and disabled. If the requirement stipulates configurable data, such as platform and system enumerations, additional test cases are written to demonstrate re-configuration of the data.

Training requirements are evaluated by demonstrating use of the simulator under anticipated operational conditions, such as the execution of a standard training scenario or loading of the system with a prescribed number of entities. Test cases may also address relevant operator manuals and maintenance training packages, although this has been outside the scope of testing previously undertaken by the authors.

5. Recent Application

The acceptance testing procedure has been applied to several training simulators and a number of technical reports written. Whilst it is inappropriate to cite specific simulator defects, there are some common, recurring problems, that are categorised below. Whilst for the most part trivial software faults, if not identified during acceptance, or whilst the simulator is under warranty, these are often far from trivial to resolve. Where defects cannot be resolved, the related simulator

functionality is ignored for training purposes, or additional equipment is installed to intercept network data exchanged between simulators, and modify or filter it accordingly.

- Measurement units are not respected, for example, knots are reported when the standard mandates metres/sec. FAULT.
- The association between objects and/or interactions is not maintained, for example between corresponding Fire and Detonation interactions. FAULT or ISSUE, depending on the association.
- Unused fields are set to zero or to random numeric values. FAULT or ISSUE, depending on circumstances.
- Packet heartbeat interval, byte ordering or data structure placement rules are not followed. FAULT.
- Enumerations, or data that may at some point need to be modified, is hard-coded into the software, and cannot be configured by operator or maintenance staff. ISSUE.
- Instability, or program crashes when fields are set to values not anticipated by the simulator. FAULT or ISSUE, depending on likelihood of a crash.

6. Conclusion

Acceptance testing is a frequently under-appreciated area of distributed simulation, as evident from its recent application. The procedure presented in this paper is beneficial to the Australian Department of Defence and the wider simulation community, as it allows network-enabled training simulators to be comprehensively tested in an efficient and repeatable manner. AOD has published a library of DIS test cases to inform project management and engineering staff alike, and this document was recently made available to the public [13]. Whilst emphasis has been placed on DIS testing, the procedure is applicable to other distributed simulation standards.

This paper details recent work undertaken in developing a uniform testing procedure for network-enabled training simulators. Development of the procedure began in 2002 during involvement with acceptance testing of a RAN operations room trainer upgrade project. It has matured substantially through the course of use, with the development of a DIS test case library representing the bulk of the effort.

7. References

[1] Zalcman, L.B., (2004), "What Questions Should I Ask Regarding DIS or HLA Interoperability For My ADF Simulator Acquisition?" Proceedings of the 9th SimTecT Conference, May 2004, Canberra, Australia.

[2] Zalcman, L.B., (2004), "Which DIS PDUs Should Be In My ADF Training Simulator?" Proceedings of the 9th SimTecT Conference, May 2004, Canberra, Australia.

[3] Valle, T., and B. McGregor, (2004), "The DMT Master Conceptual Model". Interservice/Industry Training, Simulation and Education Conference 2004, December 2004, Orlando, Florida, USA.

[4] Simulation Interoperability Standards Organization Guide for Base Object Model Use. Document no SISO-STD-003.0 DRAFT V0.12. 26 October 2005.

[5] ISO/IEC 7498-1:1994(E), (1994), "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model".

[6] IEEE 610-1990, (1990), "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries". ISBN 1-55937-079-3.

[7] Loper, M.L., (1996), "HLA Testing: Separating Compliance from Interoperability". Proceedings 14th DIS Workshop on Standards for the Interoperability of Distributed Simulations, March 1996, Orlando, Florida, USA.

[8] Ratzenberger, A., (1995), "DIS Compliant, Interoperable and Compatible: The Need for Definitions and Standards". Proceedings of the 12th DIS Workshop on Standards for the Interoperability of Distributed Simulations, March 1995, Orlando, Florida, USA.

[9] Hetzel, Bill, (1988), "The Complete Guide to Software Testing (Second Edition)". ISBN 0-89435-242-3.

[10] Simulation Interoperability Standards Organisation (SISO) Product Nomination for the IEEE 1278.1 Distributed Interactive Simulation (DIS) Product Development Group - September 2004.

[11] Ryan, P.J, Ross, P.W., Clark, P.D., and L.B. Zalcman, (2005), "Australian Contribution to International Simulation Standards Development". Proceedings of the 10th Simulation and Training Technology Conference, May 2005, Sydney, Australia.

[12] Symington, S., Kaplan, J., Kuhl, F., Tufarolo, J., Weatherly, R., and J. Nielsen, (2000), "Verifying HLA RTIs". Fall Simulation Interoperability Workshop, September 2000, Orlando, Florida, USA.

[13] Ross, P.W and P.D. Clark, (2005), “Recommended Acceptance Testing Procedure for Network Enabled Training Simulators”. Technical report no. DSTO-TR1768. Available

electronically:
http://www.dsto.defence.gov.au/publications/4301/DSTO-TR-1768.pdf

Author Biographies

PETER ROSS graduated from RMIT University in 2001 with a Bachelor of Applied Science, majoring in computer science. Mr Ross joined the Air Operations Division in 2003, and works in the Advanced Distributed Simulation Laboratory; researching communications and interoperability issues in the area of Advanced Distributed Simulation.

DR PETER CLARK is a Senior Research Scientist (Executive Level 2) in Air Operations Division, DSTO. He graduated from the Australian National University (ANU) with BSc (Hons) in 1978, and PhD in Physics in 1981. He joined the Australian Defence Department in Canberra in 1982 as a Research Scientist in the area of operations analysis. In 1989 he was appointed for a five year period as the Scientific Adviser to Army's HQ Training Command in Sydney. In 1994 he was appointed as a Senior Research Scientist with DSTO's Air Operations Division where he has worked on Navy, Air Force and Army simulation projects. Dr Clark has 25 years experience in simulation research, with an emphasis on Human-in-the-Loop real-time simulation, computer-generated forces, and the technologies associated with Advanced Distributed Simulation.