Recommended Acceptance Testing Procedure for Network Enabled Training Simulators

Peter Ross; Peter Clark
Air Operations Division
Systems Sciences Laboratory
Defence Science and Technology Organisation (DSTO)
PO Box 4331, Melbourne, Victoria, 3001, Australia
Email: peter.ross@dsto.defence.gov.au

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 Department of Defence will acquire several new platform training simulators that support distributed team training, including the Airborne Early Warning & Control operational mission simulator, AP-3C advanced flight and operational mission simulators, Armed Reconnaissance Helicopter simulator, Super Seasprite full flight mission simulator, and FFG Upgrade onboard training system and team trainer. To ensure networked interoperability between training simulators, it is essential that they be tested thoroughly against the relevant distributed simulation standards. Air Operations Division, DSTO, has been tasked to assist the testing of the aforementioned simulators. To streamline this work, a uniform procedure for testing distributed training enabled simulators has been developed. This paper defines distributed simulation concepts with regard to platform training simulators, and describes the acceptance testing procedure. Whilst the procedure is applicable to modern distributed simulation standards, emphasis has been placed on the use of Distributed Interactive Simulation and the High Level Architecture Real-time Platform Reference Federation Object Model (RPR-FOM), as the majority of new platform training simulators will employ these standards. A summary of the procedure when applied to recent training simulator acquisitions is also provided.

1. INTRODUCTION

Acceptance testing is a necessary stage in any complex procurement, as it determines whether the supplier has satisfied the requirements of the contract [1]. Over the next ten years the Department of Defence will acquire several new platform training simulators that support distributed team training, otherwise known as network-enabled training simulators. 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. However, at present there is no uniform procedure for this form of testing.

A majority of the new 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) and team trainer. 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 all existing network-enabled training simulators, including the Royal Australian Air Force AP-3C OMS, 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. As a result some simulators have a lesser technical ability to participate in distributed training exercises than others, due to both 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 [2],[3]. It is worth noting that Australia is not the only country faced with these problems, and with the advent of modern combat and mission systems, that simulator interoperability has relevance beyond training [4],[5].

This paper details recent work undertaken in developing a uniform testing procedure for network-enabled training simulators. Development of the procedure began during involvement with acceptance testing of the RAN “Maritime Warfare Training Centre” Phase 2 implementation. It has matured substantially through the course of use, with the development of a DIS test case library representing the bulk of the effort. An extract from this library is presented, as well as a summary of the procedure when applied to recent simulator acquisitions.

2. DEFINING DISTRIBUTED TEAM TRAINING

Before discussing the testing procedure, it is first necessary to identify the types of simulators being tested, and outline the role of distributed simulation, networked interoperability and acceptance testing.

2.1 Platform Training Simulator

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. Platforms, otherwise known as combat units, and tracked weapons are referred to as entities within the simulation. Whilst there are no set rules for simulator design, a generic platform training 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[1], 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 [6]. 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 the context of platform training simulators, distributed simulation is the provision of a shared virtual battlespace, in which trainees can interact. 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[2]. An object describes information that is persistent for some duration of the simulation, for example, the visual signature of a weapon. An interaction describes an instantaneous event, for example, the detonation of a weapon. Objects and interactions are parameterised by field values. Simulation model terminology varies between each distributed simulation standard, and is listed for comparison in Table 1, along with the terminology adopted by this report.

Table 1: Simulation model terminology

Adopted Term	**DIS**	**TENA**	**ALSP and HLA**
Interaction	PDU	Message	Interaction
Object	PDU with heartbeat	Stateful Distributed Object	Object
Field	Field	Attribute	Attribute or Parameter

It is important to realise that 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, can lead to non-interoperability.

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

[1] Variations include, ownairship, ownhelo and owntank. For consistency, ownship is used throughout this paper.

[2] Whilst recent distributed simulation standards boast additional modelling features, such as object inheritance, object composition and method invocation, information is effectively described through the use of interactions, objects and fields.

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. The conceptual layers of a generic distributed simulation interface for DIS, HLA and the International Standards Organisation Open Systems Interconnection (ISO/OSI) network model [7], are shown in Table 2.

Table 2: Conceptual layers and tasks of a distributed simulation interface

Layer	**DIS**	**HLA**	**ISO/OSI**
Internal model	Internal model	Simulation Object Model	Application
↕	Translation		↕
Network model	PDU types	Federation Object Model	Application
↕	Exchange		↕
Network protocol	Byte order, Structures, Heartbeats, Timeouts	Run Time Infrastructure	Presentation
			Session
Network transport	User Datagram Protocol / IP	Typically IP	Transport
			Network
			Data Link
			Physical

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 [8]. During the development of DIS and HLA, networked simulator interoperability was decomposed into three distinct levels: compliant, interoperable and compatible [9],[10].

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. 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 [11]. Unfortunately such practices are not always adopted, or if adopted, are later discarded 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) is sufficient to prevent or 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.

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 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 AOD 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 of 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).

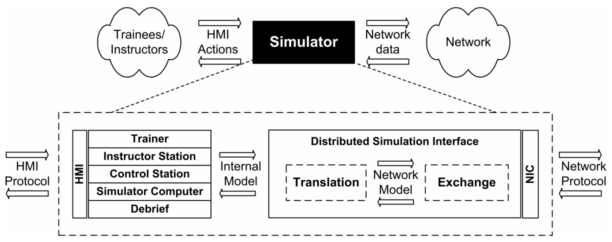

Figure 1: Black box view of a training simulator

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[3] 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.

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 a 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. AOD 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 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 3. 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

[3] Test cases are a fundamental testing concept, that identify the expected output from a specific input stimulus. A test case is considered to pass if the output witnessed during the test execution meets the output expected.

a clear and consistent understanding of the standards requirements. In related research, AOD has documented known ambiguity and established interpretations of the DIS standard, and is actively involved in the development of a revised standard [12],[13]. To add authority and assist defect resolution, test cases should reference the original requirements text, and state any interpretations applied.

Table 3: Typical distributed simulation requirements

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

Network transport and hardware requirements are normally tested 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 [14].

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, for example, 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.

A standard test case specification format was assembled, based on existing test documentation standards [15]. An extract from AOD's DIS test case library is shown in Table 4. Related tests cases are grouped into tables, with columns that describe the test case identification number, execution requirement (M=mandatory, S=applied to all subsequent tests), applicable simulator components (T=trainer, C=control station, and so on), the test input and expected output, and pass/fail criteria (R=requirement, D=desirable). Fields are highlighted in italics, bitfields are underlined, and enumerated value names are wrapped in single quotes.

Table 4: Extract of the IFF test case group (send testing)

ID	E	C	Input (HMI)	Expected Output (NIC)	P
S-5.0	S	-	(any – IFF Layer 1)	IFF PDUs are sent at a 10 second heartbeat rate, or when one or more operational parameters has changed and the two second change latency has elapsed. [IEEE 1278.1A, section 4.5.6.5.2] *System Type*, *System Name* and *System Mode* are defined in SISO-EBV section 8.3.1.1, and indicate an appropriate IFF transponder device. [IEEE 1278.1A, section 5.2.58] The _*Change Indicator*_ bit of *Change/Options* is set to 'Initial report or change since last report' or 'No change since last report'. [IEEE 1278.1A, section 4.5.6.5.2] The _*Layer 1*_ bit of *Information Layers* is set to 'On', and _*Layer 2*_ bit of *Information Layers* is set to 'Off'. [SISO-EBV 8.3.2.2.10] *Antenna Location wrt Entity* indicates the location of the transmitter antenna relative to the ownship entity location. [IEEE 1278.1A, section 5.3.7.4.1] If one or more modes are enabled, The _*System On/Off*_ bit of *System Status* is set to 'On' and the _*Operational Status*_ bit is set to 'Operational'. [SISO-EBV, section 8.3.6.1]	R
S-5.1	M	C	Create the ownship entity.	N/A	
S-5.2	M	T	Activate IFF transponder with, no modes enabled, and wait at least 15 seconds.	The _*Status*_ bit of *Parameter 1* through *Parameter 6* is set to 'Off'. [SISO-EBV, section 8.3.6.1]	D
S-5.3	-	T	Enable Mode 3 with code '2345', for at least 15 seconds.	The _*Status*_ bit of *Parameter 3* is set to 'On', the _*Damage*_ bit is set to 'No Damage' and the _*Malfunction*_ bit is set to 'No Malfunction'. [SISO-EBV, section 8.3.6.1] The _*Code Element*_ bits of *Parameter 3* indicate '2345'. [SISO-EBV, section 8.3]	R
S-5.4	M	T	Deactivate IFF transponder and wait at least 15 seconds.	IFFPDUs are no longer sent for the ownship entity, or one or more IFFPDUs are sent with the _*System On/Off*_ bit of *System Status* set to 'Off'. [SISO-EBV, section 8.3.6.1]	R

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, reoccurring problems, and these have been categorised below. Whilst for the most part trivial software faults, if not identified during acceptance, or whilst the simulator is under warranty, they are often far from trivial to resolve. Where defects cannot be resolved, the related simulator functionality is ignored for training purposes, or filtering 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. OTHER SIMULATIONS STANDARDS

Although the procedure was originally intended for DIS standards testing, it is independent of the underlying distributed simulation technology, and could be applied to other standards, such as HLA, were the need to arise. Test case development, however, will be required to address specific requirements of the standard.

Much the existing DIS test library can be reused, if the standard employs a network model that is equivalent to the DIS network model. For example, adapting the earlier IFF test case example to the HLA Real-time Platform Reference Federation Object Model would require modification of tests S-5.0 and S-5.4, in order to address the expected outputs that are specific to HLA. Minor wording changes would also be necessary, for example, *Code Element* would become *Mode3ACode.*

7. 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 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. Over the next 24 months, AOD will apply this procedure to several new training simulator acquisitions, and intends to publish its library of DIS test cases to inform project management and engineering staff alike. Whilst emphasis has been placed on DIS testing, the procedure is applicable to other distributed simulation standards.

REFERENCES

1. Defence Material Organisation, (2004), "Defence Procurement Policy Manual - Version 5.0".
2. 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.
3. Zalcman, L.B., (2004), "Which DIS PDUs Should Be In My ADF Training Simulator?" *Proceedings of the 9th SimTecT Conference*, May 2004, Canberra, Australia.
4. Valle, T., and B. McGregor, (2004), "The DMT Master Conceptual Model". *Interservice/Industry Training, Simulation and Education Conference 2004*, December 2004, Orlando, Florida, USA.
5. Byrum, C., Mittura, A., and M. Hohneker, (2000), “Distributed Engineering Plant Simulation/Stimulation Environment Accreditation of Accuracy”. *Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications*, August 2000, San Francisco, California, USA.
6. Schaafstal, A.M., Lyons, D.M., and R.T. Reynolds, (2001), "Teammates and Trainers: The Fusion of SAFs and ITSs". *Proceedings of the 10th Conference on Computer Generated Forces and Behavioural Representation*, May 2001, Orlando, Florida, USA.
7. ISO/IEC 7498-1:1994(E), (1994), "Information technology - Open Systems Interconnection - Basic Reference Model: The Basic Model".
8. IEEE 610-1990, (1990), "IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries". ISBN 1-55937-079-3.
9. 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.
10. 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.
11. Hetzel, Bill, (1988), "The Complete Guide to Software Testing (Second Edition)". ISBN 0-89435-242-3.
12. Simulation Interoperability Standards Organisation (SISO) Product Nomination for the IEEE 1278.1 Distributed Interactive Simulation (DIS) Product Development Group - September 2004.
13. 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.
14. 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.
15. IEEE 829-1998, (1998), "IEEE Standard for Software Test Documentation". ISBN 0-7381-14448.