CDMTS: A Common User Interface for Multiple Training Environments

Bela Joshi, Richard King, and Brian Teer
Alion Science and Technology Corporation, BMH Operation
Norfolk, VA 23513

bjoshi@alionscience.com, rking@alionscience.com, bteer@alionscience.com

ABSTRACT

A recent Department of Defense goal is to achieve a seamlessly integrated distributed training environment. Such an environment would integrate Live, Virtual and Constructive (LVC) simulation assets to provide realistic training to the warfighter. However, building such a complex environment presents unique challenges. Technical challenges arise from the disparate platforms, technologies, and protocols used by the simulation assets. Training challenges arise from the increased cognitive demands of simultaneously managing different Instructor Operator Stations (IOS).

A common IOS (C-IOS) theoretical framework for an integrated distributed training environment was developed by NAVAIR TSD. The framework specifically addressed cognitive work load, IOS training, distributed mission training, and acquisition issues. This C-IOS concept was implemented as the Common Distributed Mission Training System (CDMTS). CDMTS has been deployed in various training environments and has become a common tool for integrating and managing Modeling & Simulation (M&S) training technologies.

In this paper, we describe the challenges associated with implementing training tools such as CDMTS, which integrate diverse technologies and simulation environments. We include benefit and tradeoff considerations for issues such as common data specifications, semantic and functional interoperability, and distributed architectures. Further, we present lessons learned from work done to incorporate CDMTS into multiple training environments.

ABOUT THE AUTHORS

Bela Joshi is a Senior Software Engineer in the Alion BMH Operation Applications Technology Division. She is the Project Manager of the CDMTS development team. Bela received a PhD in Engineering Management from Old Dominion University, Norfolk, VA.

Richard King is a Research Software Engineer in the Alion BMH Operation Applications Technology Division. He served as the Technical Lead of the CDMTS development team. Richard received a BS degree in Computer Science from Virginia Tech.

Brian Teer is the manager of the Alion BMH Operation Applications Technology Division. He is also the Senior Project Manager of the CDMTS development team and has been involved in CDMTS since its inception. Brian received a MS in Computer Science from North Carolina State University**.**

CDMTS: A Common User Interface for Multiple Training Environments

Bela Joshi, Richard King, and Brian Teer
Alion Science and Technology Corporation, BMH Operation
Norfolk, VA

bjoshi@alionscience.com, rking@alionscience.com, bteer@alionscience.com

INTRODUCTION

The use of modeling and simulation in training today's warfighters is becoming increasingly common. A survey of military training simulation environments reveals the numerous technologies, architectures, protocols, and platforms that are employed by such simulations. An overarching goal of the US Department of Defense (DoD) is to develop a training environment integrating various Live, Virtual and Constructive (LVC) simulation assets (Thorp, 2003; Bizub and Phillips, 2004). The demands of a distributed training environment significantly increase the cognitive workload of instructors. An instructor has to simultaneously interact with multiple operator stations based on different platforms, architectures, and Graphical User Interfaces (GUIs). A common Instructor Operator Station framework (C-IOS) that addresses the issues of cognitive work load based on distributed mission training has been developed (Walwanis Nelson et al., 2003). This framework is based on principles of Human Factors Engineering and specifically addresses all aspects of training, including scenario development, exercise monitoring and control, performance measurement, and brief/debrief (Owens 2003, Stiso et al, 2004).

The implementation of a common IOS for the distributed simulation environment is not without its share of technical challenges. These challenges can be attributed to the need for achieving semantic and functional interoperability among diverse applications, architectures, protocols, and platforms. A survey of IOS platforms reveals Boeing's Common IOS, which consists of an extensible software framework of configurable components. Boeing's Common IOS is a user interface for several standalone simulators/trainers (F-15C Commercial Training Simulator Services Distributed Mission Operations trainer; the T-38 Operation Flight Trainer and Unit Training Device; the AH-64D Longbow Crew Trainer, Full Mission Simulator, and Field Deployable Simulator). Although Boeing's Common IOS is extensible and configurable, it is proprietary.

This paper presents the technical implications of implementing the C-IOS framework as the Common Distributed Mission Training System (CDMTS). CDMTS is government-owned open source software that serves as a common IOS for multiple simulation applications in a distributed network. Design considerations and technical challenges that are commonly associated with the implementation of such integrated training tools are described. In addition, lessons learned from deploying CDMTS in multiple training platforms and user communities are discussed.

Sharing technical information about emerging tools and technology such as CDMTS contributes to the training community and the overall DoD goal of an integrated simulation training environment.

CDMTS OVERVIEW

CDMTS is based on a theoretical framework for a C-IOS based on Human Factor Engineering principles (Walwanis-Nelson, 2003). The framework is empirically tested by verifying capabilities and user interfaces with military service users. The C-IOS specifically addresses cognitive workload, IOS training, distributed mission training, and acquisition costs.

The key capabilities of CDMTS are as follows:

1. Integrates virtual and constructive assets in a distributed simulation network
2. Serves as a common user interface for multiple constructive simulation applications
3. Capable of performing planning, monitoring, control, and after action review of an exercise
4. Government-owned, open source, extensible framework can be customized for specific deployments
5. Learning science based user interface is proven through empirical testing

Figure 1 contains a screen shot of CDMTS being used to monitor an exercise.

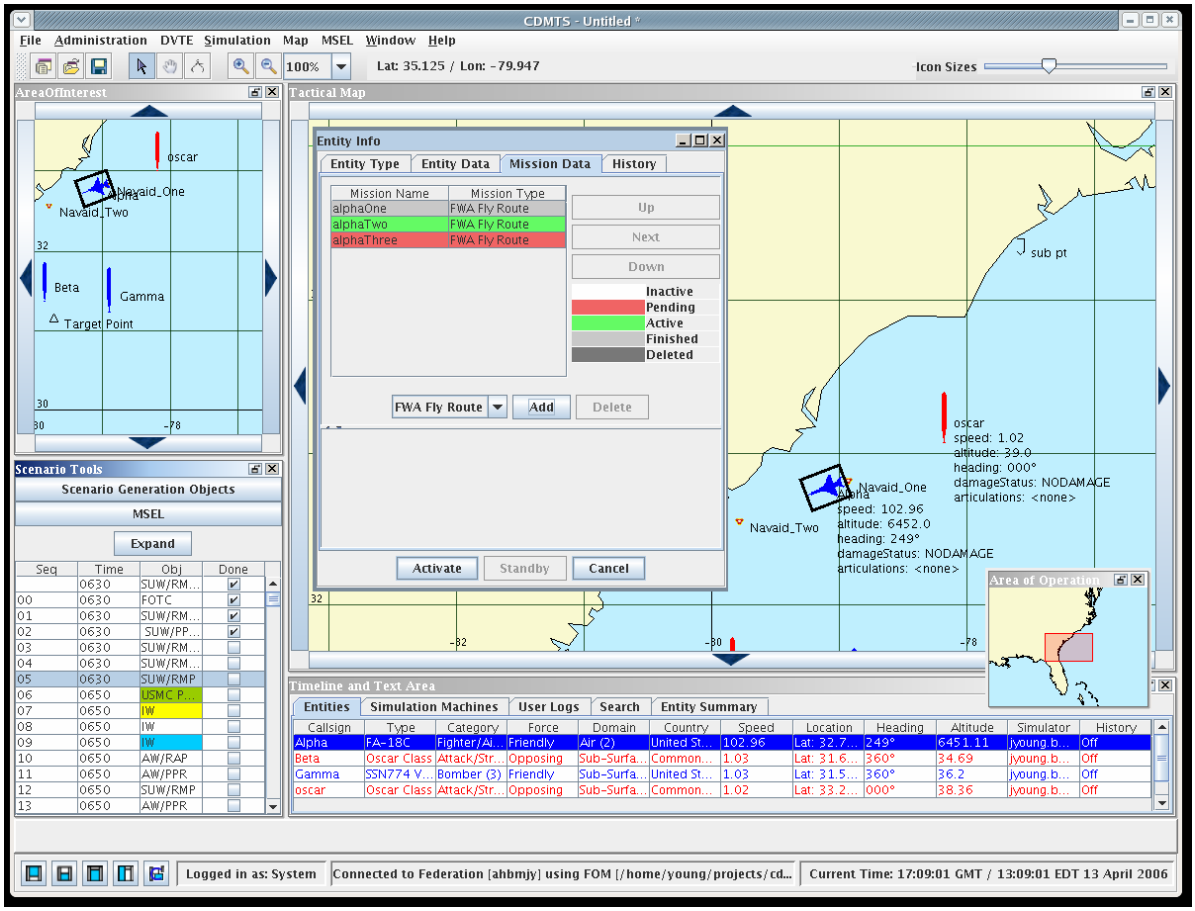

Figure 1. CDMTS Showing an Exercise in Progress

DESIGN CONSIDERATIONS

Several technical challenges need to be addressed in designing and implementing a training tool that integrates diverse simulation technologies. The issues of common data specifications and semantic as well as functional interoperability are of particular concern. This section describes these design considerations as well tradeoff and benefit analyses.

Communication Protocol

Adopting an appropriate communication protocol is an important design consideration for a common IOS tool that integrates applications running under different simulation architectures. Candidate communication protocols that were considered for CDMTS include the High Level Architecture (HLA) and Distributed Interactive Simulation (DIS). In addition, commercial industry standards such as Common Object Request Broker Architecture (CORBA) and Remote Method Invocation (RMI) were also considered.

HLA and DIS are the two dominant distributed simulation interoperability standards used in military training. (To a lesser extent, the Test and Training Enabling Architecture (TENA) architecture is also used, primarily where live platforms and/or ranges are involved.). Among these common DoD standards, HLA is generally considered superior to DIS (Buss and Jackson, 1998). In addition, HLA is able to support a range of functional areas, including training, analysis, and systems acquisition.

Distributed computing technologies in the commercial sector are also able to achieve interoperability in a distributed environment. Two of the commonly used technologies are CORBA and RMI (Buss and Jackson, 1998). CORBA offers a fairly well established and mature collection of specifications and protocols for application interoperability, and ensures independence

of platforms, operating systems, and programming language (Object Management Group, 1998; Orfali and Harkey, 1998). RMI is part of the standard Java toolkit and is platform independent (Javasoft, 1997). RMI uses Java's own interface syntax as its object interface language, and consequently is simple to implement. RMI is ideally suited for the implementation of new Java based software systems that do not depend on legacy code.

While CORBA and RMI are general purpose distributed computing architectures, HLA is targeted specifically towards distributed simulation applications (US DoD 1996; US DoD 1998). For example, HLA has the distinct advantage of possessing the Time Management Service, which supports the notion of a simulation clock. HLA also supports transfer of object ownership, whereby an aircraft modeled in one application can carry missiles that are simulated by another application. In addition, HLA offers distinct benefits when multiple legacy simulation models written in different languages are involved (Buss and Jackson, 1998).

Due to these reasons, HLA was adopted as the communication protocol.

Functional Interoperability

Another design consideration that needs to be addressed in implementing an integrated training tool for distributed simulation technologies is the ability to communicate with other simulation applications. In CDMTS, this communication is facilitated by the use of HLA and a CDMTS Base Object Model (BOM). CDMTS uses HLA communication in two manners. It joins an HLA federation and consumes data in order to display federation data in its GUI. CDMTS also uses HLA and the CDMTS BOM to convey data to other simulation applications in the training exercise. The key features of the communication model to achieve functional interoperability are:

1. Each CDMTS-compliant application must implement the CDMTS Simulation Communications Interface (SIMCI) to enable CDMTS communication.
2. Non-HLA protocols used in the training environment must be converted to HLA via appropriate gateways in order for CDMTS to receive and use the data.
3. Native simulation application communications must be unaffected by CDMTS. For example, CDMTS-compliant DIS-based applications continue to send DIS Protocol Data Units (PDUs), and HLA-based applications continue to send HLA objects/interactions using their native interfaces. All communication with CDMTS occurs through the application's SIMCI interface.

Figure 2 shows a conceptual view of the CDMTS communication model architecture (Alion Science and Technology, Technical Report, 2006).

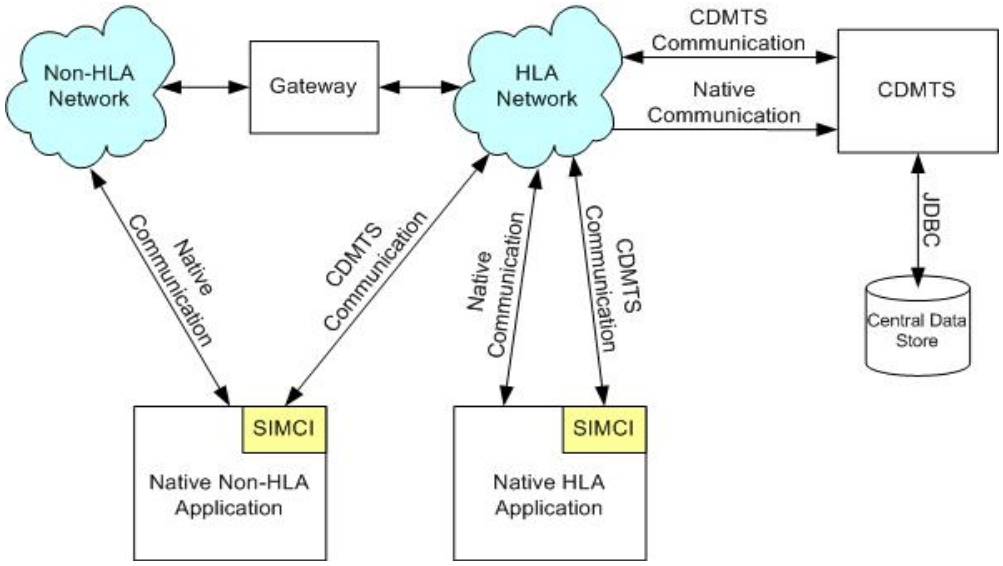

Figure 2. CDMTS Communication Model

As mentioned above, HLA based communication is used for exchanging data with other simulation applications. Data exchange is facilitated through a BOM that defines objects and interactions to be

exchanged. These objects and interactions are described in the BOM in terms of attributes. Attributes are generally of simple or complex data type; i.e. a data type comprised of other simple or complex data types.

To facilitate the communication of CDMTS specific data to other applications, a suitable BOM was defined. In keeping with the goal of achieving flexibility and extensibility, the CDMTS BOM was designed to simplify attribute specification. The BOM avoids defining specific and detailed attributes for each object and interaction. Instead, attributes are described very simply as data strings containing all relevant details in an Extensible Markup Language (XML) formatted string. Figure 3 shows a graphical view of an excerpt of the CDMTS BOM. Figure 4 depicts a tabular representation of an excerpt of the BOM.

A direct benefit of this approach is that the BOM remains stable as the data passed inside its objects and interactions evolves with the evolving capabilities of CDMTS. In addition, native HLA interfaces ignore CDMTS specific BOM traffic. A tradeoff is that each receiver is responsible for validating and parsing the attribute XML data. It was decided that this was an acceptable tradeoff, as only the CDMTS SIMCI components would be affected.

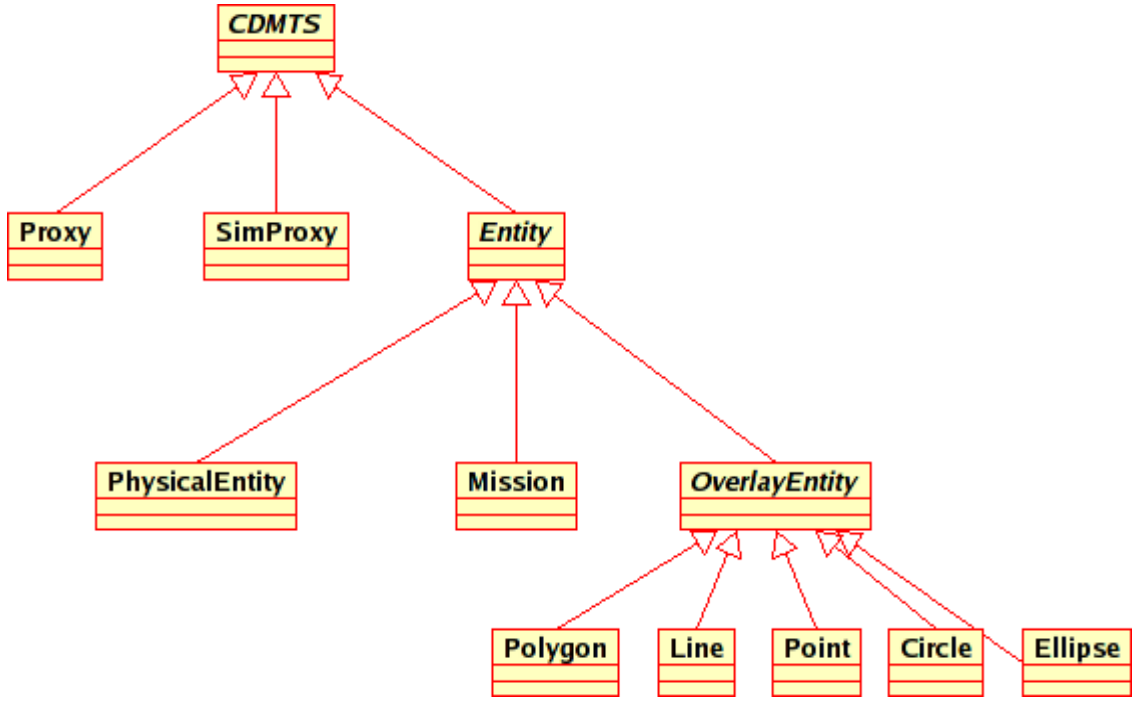

Figure 3. A Graphical Depiction of an Excerpt from the CDMTS BOM

Entity		
Definition	The base class for entity classes published by CDMTS	
Attributes	ObjectId	Unique RTI object identifier
	ProxyName	Identifies the CDMTS instance that created this object
	SimProxyName	Identifies the simulation application that is or will be responsible for simulating or using this object

PhysicalEntity		
Definition	Represents an entity simulated by a simulation application	
Attributes	Data	An XML formatted string that describes the entity data (e.g., location, fire permission, orientation, speed, etc.)

Mission		
Definition	One or more tasks that were created and assigned to an entity in CDMTS	
Attributes	EntityId	The identifier of the associated entity
	NumberOfTasks	The number of tasks assigned to the mission
	Tasks	An XML formatted string that describes the data (e.g., task type, target, etc.) specified for each task assigned to the mission. A mission can have one or more tasks

OverlayEntity		
Definition	The base class for overlay entity classes published by CDMTS	
Attributes	Name	The overlay entity name
	Data	Contains an XML data string

Figure 4. A Tabular Representation of an Excerpt from the CDMTS BOM

Common Data Specification

In building an integrating tool it is important to standardize data specifications that are applicable across various simulations and applications. In recent years, XML has become the standard for exchanging data across multiple platforms, languages, and applications, and it enjoys strong industry support (W3C Consortium Recommendation, 2000). In addition, adoption of XML has the potential to facilitate future integration with emerging data standards technologies, such as Extensible Battle Management Language (XBML) (Hieb, 2003).

For all of these reasons, XML was adopted as the data specification standard. HLA object and interaction attribute data, scenario and mission planning data, as well as CDMTS configuration files use XML technology.

Programming Language

Simulation applications and environments typically span the Windows and Linux platforms. In addition, standalone simulators or trainers may have custom hardware and proprietary operating systems. The Java programming language offers the unique advantage of being platform independent and is hence ideally suited for an integrated simulation environment framework. Advantages of Java include the following:

1. Platform independent programming language

2. Built in support for GUIs
3. Object oriented programming language
4. Automatic garbage collector to manage memory
5. Native support for security
6. Strong open source support

Java has been likened to a "platform," due to its impressive library containing reusable code and an execution environment that provides services such as security, portability across operating systems, and automatic garbage collection (Hortsmann and Cornell, 2004). For the above reasons, the Java programming language was selected.

Semantic Interoperability

Generic mission planning in an environment-independent manner is a key interoperability area for an integrated training tool. The challenge is to generate a single mission plan that can be understood by multiple simulation applications. Different simulation applications require inputs in different formats for creating and tasking the same entity. In some cases, they may require different data altogether.

One approach to generic mission planning is to build awareness of the capabilities of each simulation application in CDMTS. The advantage of this approach is that communication data is always passed in the correct content and format to the simulation application. However, a drawback is that scenarios and mission plans are specific to the platform used in a particular deployment; and re-use is limited. Thus a new scenario or mission plan will have to be generated for a new environment.

An alternate approach is to generate a "one format fits all" type of scenario or mission plan. This ignores the specific capabilities of each simulation application. The main advantage of this approach is that the scenarios can be reused across multiple environments. The main drawback is that the CDMTS communications component that resides in a simulation application has to perform rigorous data validation for data content and format.

In the case of CDMTS, the "one format fits all" approach with its significant advantage of applicability across multiple training environments was adopted.

LESSONS LEARNED FROM MULTIPLE DEPLOYMENTS

A C-IOS tool such as CDMTS provides the majority of IOS functionality, i.e., scenario or exercise planning, monitoring, control, and after action review. In addition to these common features, each deployment may require additional features based on the specific nature and role of the deployment. Examples of these CDMTS deployments are listed below.

1. **Navy Aviation Simulation Master Plan (NASMP) F-18 test bed** – CDMTS is the instructor operator station and is integrated with Joint Semi Automated Forces (JSAF), Next Generation Threat System (NGTS), and F-18 simulator via the Federation Object Model (FOM) bridge.
2. **Anti Submarine Warfare (ASW) Virtual At Sea Training (VAST) Mission Rehearsal Tactical Team Training (MRT3)** – CDMTS is the IOS and is integrated with JSAF. CDMTS is also used to launch the various applications such as Pilot Station (Microsoft Flight Simulation), Air Tactical Officer (ATO), and Sensor Operator (SENSO).
3. **Virtual Fire Support Trainer (VFST) Deployable Virtual Training Environment (DVTE)** – CDMTS is used to configure various applications (for example, Caber, Marine Digital Voice (MDV), and JSAF) to machines on the network. It is also used to launch and control these applications.
4. **Virtual Technologies and Environments (VIRTE)** – CDMTS is used configure and launch applications, such as Forward Observer Personal Computer Simulator (FOPCSim), Forward Air Controller Personal Computer Simulator (FACPCSim), Dismounted Infantry Virtual After Action Review System (DIVAARS), and MDV.
5. **Multi-purpose Supporting Arms Trainer (MSAT)** – CDMTS back end component SIMCI is used to task JSAF.
6. **MH-60R Tactical Operational Flight Trainer (TOFT)** – CDMTS is planned to be an element of the MH-60 R IOS and will integrate with JSAF and the MH-60 R trainer.
7. **U.S. Joint Forces Command's Joint Advance Training Technology Laboratory (JATTL)** – CDMTS has been used as a viewer station in large scale exercises involving various virtual and constructive simulations.

It is clear from these examples that CDMTS can potentially perform different roles in different training environments. These deployments offer several opportunities for lessons learned; and some of these are presented below.

Extensible and Customizable Architecture

A desirable feature of an integrated training tool such as CDMTS is the ease with which custom functionality can be added for each specific deployment. In order to support this requirement, the CDMTS architecture is evolving towards a plug-in architecture. A plug-in architecture lets developers build a master application from constituent components.

Under this scheme, major functionality that is required across deployments will be supplied by the CDMTS core engine. Examples of such core functionality include planning, execution, control, after action review, and performance measurement. Auxiliary functionality (specific to a particular deployment) can be provided by a specially built plug-in component. Examples of auxiliary functionality would be displaying range rings and splash points that are required in an ASW training environment.

Currently, work is in progress to re-architect CDMTS for the plug-in architecture. This includes publishing a well defined Application Program Interface (API) or a contract for CDMTS plug-ins.. Future plug-in components can then be built in compliance with the specified contract, and can be simply configured and loaded at run-time.

Configuration

Deploying for multiple environments requires the ability to configure CDMTS appropriately for each specific platform. For example, the entities, tasks, and GUI components required for an ASW VAST MRT3 deployment may be quite different than for a VFST DVTE deployment. To aid in the ease of deployment, the CDMTS architecture specifies configuration information in external XML data files as opposed to hard-wiring in code. This approach ensures that code is relatively stable and does not have to be changed to accommodate configuration for each new deployment environment. Thus the code base is generic and reusable. In addition, deployments become more manageable, as making changes to an XML data file during integration is easier than changing code.

Usability Analysis and Feedback

Since CDMTS is deployed in multiple environments and platforms, one of the on-going challenges is to ensure that the GUI continues to satisfy the unique training needs of numerous user communities. To ensure that CDMTS continues to be an intuitive and effective tool, usability studies are conducted on a regular basis. These studies are aimed at specific user communities and gather feedback data from subject matter experts in various training environments. Results of the usability studies are analyzed and incorporated when implementing new features as well as enhancing existing features. As its user base grows, this approach has proven invaluable in ensuring the effectiveness of CDMTS.

Rapid Development Environment

In order to build new features for multiple deployments as well as to solicit early usability feedback, it is imperative to produce well tested code in short iteration cycles. The use of Java programming language has proven to be of great benefit in this area. Compared to other programming languages, Java arguably has the best tool support that includes open source libraries, Integrated Development Environments (IDE), as well as unit and functional testing. All of these features have a significant impact on enabling CDMTS developers to produce new features in relatively short amounts of time.

CONCLUSIONS

CDMTS has been successfully implemented and deployed as a common IOS in multiple distributed simulation environments. Its theoretical framework is based on principles of learning science and has been empirically tested for training station design. Due to its adaptable and extensible nature, CDMTS has become a common tool for integrating and managing diverse training environments. The design considerations, technical challenges, and lessons learned that are presented here are equally applicable to other tools for integrating diverse distributed training systems and simulation applications. The authors believe that this paper contributes to the training community as well as literature devoted to the ongoing DoD quest for a seamless distributed simulation training environment.

ACKNOWLEDGEMENTS

The views expressed herein are those of the authors and do not necessarily reflect the official position of the organizations with which they are affiliated. The authors would like to thank the program managers who have supported this work.

REFERENCES

Alion Science and Technology Corporation, BMH Operation. (2006). Technical Report: Architecture Description Document for the Common Distributed Mission Training System. Version 0.2.

Bizub, W. and Phillips, M. (2004). Correcting the vision – Introducing the Joint National Training Capability (JNTC) Advanced Training Technology Laboratory (JATTL) [CD-ROM]. Proceedings of the 2004 Interservice/Industry Training, Simulation, and Education Conference, Orlando, FL, 137-147.

Buss, A. and Jackson, L. (1998). Distributed Simulation Modeling: A comparison of HLA, CORBA, and RMI. Proceedings of the 1998 Winter Simulation Conference, 819-825.

Farley, J. (1997). Java Distributed Computing, O'Reilly, Cambridge, MA.

Hieb, M. (2003). Extensible Battle Management Language: XBML Presentation to DMSO C4I-SimTEM. http://netlab.gmu.edu/XMSF/pdfs/XBML3Oct03.pdf

Hortsmann, C. and Cornell, G. (2004). Core Java. Volume 1-Fundamentals, Sun Microsystems Press.

Javasoft, Inc. (1997) Remote Method Invocation Specification.

Object Management Group (1998). The Common Object Request Broker: Architecture and Specification.

Orfali, R and Harkey, D. (1998). Client/Server Programming with Java and CORBA, John Wiley and Sons, Inc, New York, NY.

Owens, J.M. (2003). Design issues and considerations for a Common Instructor Operator Stations (IOS) in support of the Naval Aviation Simulation Master Plan (NASMP) (CHI Systems Technical Report 030225.0001-0028). Lower Gwynedd, PA: CHI Systems, Inc.

Stiso, M., Owens, J., Fowlkes, J., Eitelman, S., and Hafich, A. (2004). Common Instructor Operating Station (C-IOS) requirements analysis: Scenario development and computer-generated forces integration (CHI Systems Technical Report 031107.0001-0034). Lower Gwynedd, PA: CHI Systems, Inc.

Thorp, H, (2003). Training Transformation and the Joint National Training Capability. Brief by the Director of the JNTC Joint Management Office.

US Department of Defense (1996). High Level Architecture Object Model Template, IEEE P1516.2

US Department of Defense (1998). High Level Architecture Interface Specification, Version 1.3 of IEEE P1516.1, M&S HLA – Federate I/F Spec, DRAFT 1 of 20 April 1998.

W3C Consortium Recommendation, (2000). Extensible Markup Language (XML) 1.0. Second Edition. http://www.w3.org/TR/2000/REC-xml-20001006

Walwanis Nelson, M. M., Lackey, S., & Bryan, D. (2005). Developing a common mission training station for distributed simulation: The performance measurement challenge. *Proceedings of the 1st annual Training, Education, & Simulation International (TESI) Conference*, Maastricht, The Netherlands.

Walwanis-Nelson, M., Owens, J., Smith D., and Bergondy-Wilhelm, M. (2003). A Common Instructor Operator Station Framework: Enhanced Usability and Instructional Capabilities. Proceedings of the 2003 Interservice/Industry Training, Simulation and Education Conference.