NISTIR 7131

PC Modeling and Simulation Guidelines: Volume 1 - Overview

Charles McLean
Frank Riddick

Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

Prepared for:

Naval Education and Training Command
Learning Strategies Division
250 Dallas Street
Pensacola, Fl 32508

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

NISTIR 7131

PC Modeling and Simulation Guidelines: Volume 1 - Overview

Charles McLean
Frank Riddick

Manufacturing Systems Integration Division
Manufacturing Engineering Laboratory

August 2004

U.S. DEPARTMENT OF COMMERCE
Donald L. Evans, Secretary
TECHNOLOGY ADMINISTRATION
Phillip J. Bond, Under Secretary of Commerce for Technology
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
Arden L. Bement, Jr., Director

PC Modeling and Simulation Guidelines:
Volume 1 - Overview

August 2004

Prepared for:

Naval Education and Training Command
Learning Strategies Division
250 Dallas Street
Pensacola, Fl 32508

Prepared by:

National Institute of Standards and Technology
Charles McLean – Manager
Frank Riddick – Computer Scientist
Simulation and Visualization Program

Preface

The PC Modeling and Simulation Guidelines are a series of documents that have been developed under the direction of the Naval Education and Training Command, Pensacola, Florida. The object of the guideline documents is to establish a consistent and cost-effective approach for Navy simulation-based training development efforts. The documents are focused on addressing training applications that employ simulation technology on personal computer platforms, as opposed to those that require special purpose simulator hardware.

The first documents in the PC Modeling and Simulation Guidelines series are:

Volume 1: Overview
Volume 2: PC Simulation – A Decision Process

Other volumes that are in development, or are planned, provide more details on topics introduced in the Overview document. Topics to be addressed in future documents include: an architecture for a generic open-source video game engine for training, game engine module interface specifications, a model of Navy PC simulation-based training needs, a re-usable learning object classification scheme, and software testing procedures.

The Guidelines will undoubtedly evolve over time as Navy needs and technology changes. Feedback in the form of corrections or comments is sincerely appreciated. Recommendations concerning the establishment of additional guidelines are also welcomed.

Disclaimer: Work described in this report was sponsored by the Naval Education Training Command, Pensacola, Florida, and the National Institute of Standards and Technology (NIST), Gaithersburg, Maryland. No approval or endorsement of any commercial product by the Navy or National Institute of Standards and Technology is intended or implied. The work described was funded by the United States Government and is not subject to copyright.

Acknowledgements: The authors would like to acknowledge the many contributions of and guidance from Mike Cleveland in the creation of this report. Although the authors have tried to reference or acknowledge the contributions of others wherever possible, some source materials have been drawn from earlier draft government reports whose authors were not identifiable.

Table of Contents

Preface **2**
Table of Contents **3**
1. Introduction **6**
1.1 Why is PC simulation-based learning important to the Navy? 7
1.2 Relevant Studies and Policies 8
1.3 Goals and Objectives 9
2. Terminology, Categorization, and Criteria **11**
2.1 PC Simulation-based Learning Terminology 11
2.1.1 Instruction and Learning Modes 11
2.1.2 Content Management, Delivery, and Tracking Systems 12
2.1.3 Simulation Model Types 13
2.1.4 Simulation System Elements 14
2.1.4.1 Browser Elements 14
2.1.4.2 Simulation Engine Elements 15
2.1.4.3 Distributed Simulation Elements 16
2.1.4.4 Simulation-based Learning Elements 17
2.1.5 Quality of the Simulation Experience 17
2.1.5.1 Fidelity 18
2.1.5.2 Interactivity 18
2.1.5.3 Immersion 19
2.2. Categorization Scheme of Simulation Types 20
2.2.1 Cognitive Support Simulation 20
2.2.2 PC Software Simulation 20
2.2.3 Situational Simulation 21
2.2.4 Procedural Simulation 21
2.2.5 Virtual World Simulation 21
2.3 Suggested Simulation Suitability Criteria 22
2.3.1 Learning Levels 22
Table 2-1 Sample Learning Levels for PC Simulation Consideration 23
2.3.2 Training Applications and Simulation Types 24
Table 2-2 Types of Training Benefited by PC Simulation 24
2.3.3 Evaluation Methodology 24
Table 2-3 Benefit Considerations for ROI Calculations 26
3. PC Simulation-Based Learning Architecture **27**
3.1 Requirements Summary 27
3.2 Architectural Elements 28
3.2.1 Learning Content Management System (LCMS) 29
3.2.2 Learning Management System (LMS) 31
3.2.3 Course Management System (CMS) 31
3.2.4 Student Learning Platform – PC (SLP-PC) 32
3.2.5 Student Learning Platform – PDA (SLP-PDA) 32
3.2.6 Web Browsers (WB) 33
3.2.7 Web Browser Plug-ins (WBPI) 33
3.2.8 Stand-alone Simulator (SAS) 34

3.2.9 Distributed Simulation Component (DSC) 34
3.2.10 Courseware Authoring System (CAS) 35
3.2.11 Course Testing System (CTS) 35
3.2.12 System Interfaces 36
4. Simulation-based Training Using Game Technology 38
4.1 Training System Vision 38
4.2 Example Training Applications 40
4.3 Key Game System Elements 40
4.3.1 Graphics and Sound 41
4.3.2 Game Engine Management 41
4.3.3 Character Animation 42
4.3.4 Physics 42
4.3.5 Artificial Intelligence 43
4.3.6 Game Editor and Development Tools 43
4.3.7 Application Data Import and Export 43
4.3.8 Multi-Player Operations and Server Support 43
4.3.9 Software Distribution and Security Mechanisms 44
4.3.10 Learning System Support 44
4.4 Development Approach 45
4.4.1 Open Source Code 45
4.4.2 Development Process Steps 47
5. Standards 50
5.1 Standards Organizations 50
5.1.1 Advanced Distributed Learning (ADL) Initiative 51
5.1.2 American National Standards Institute (ANSI) 51
5.1.3 Aviation Industry CBT (Computer-Based Training) Committee (AICC) 51
5.1.4 Department of Defense (DoD) 52
5.1.5 IMS Global Learning Consortium, Inc. (IMS) 52
5.1.6 Institute of Electrical and Electronics Engineers, Inc. (IEEE) 52
5.1.7 International Organization for Standardization (ISO) 53
5.1.8 National Institute of Standards and Technology (NIST) 53
5.1.9 Organization for the Advancement of Structured Information Standards (OASIS) 54
5.1.10 Object Management Group (OMG) 54
5.1.11 Synthetic Environment Data Representation and Interchange Specification (SEDRIS) 54
5.1.12 Simulation Interoperability Standards Organization (SISO) 55
5.1.13 Web 3D Consortium (Web 3D) 56
5.1.14 World Wide Web Consortium (W3C) 57
5.2 Standards Specifications 58
5.2.1 Extensible Markup Language Standards 58
5.2.1.1 Extensible Markup Language (XML) 58
5.2.1.2 Department of the Navy (DON) XML 59
5.2.1.3 SEDRIS 59
5.2.1.4 Shareable Content Object Reference Model (SCORM) 59
5.2.2 Simulation Standards and Specifications 60
5.2.2.1 High Level Architecture (HLA) 60

5.2.2.2 Extensible Modeling and Simulation Framework (XMSF) 61
5.2.2.3 Discrete Event System Specification (DEVS) 61
5.2.3 Multimedia Standards 61
5.2.3.1 Virtual Reality Modeling Language (VRML)/X3D 61
5.2.3.2 H-Anim 62
5.2.4 Product Representation Standards 63
5.2.4.1 Standard for the Exchange of Product Model Data (STEP) 63
5.2.4.2 Initial Graphics Exchange Specification (IGES) 63
5.2.5 Knowledge Representation Standards 64
5.2.5.1 Knowledge Interchange Format (KIF) 64
5.2.5.2 Web Ontology Language (OWL) 64
5.2.6 Mobile Code Programming Language Standards 64
5.2.6.1 Java 65
5.2.6.2 JavaScript/Jscript/ECMAScript 65
5.2.6.3 VBScript 65
5.2.6.4 Perl 65
5.2.6.5 PHP: Hypertext Preprocessor (PHP) 65
5.2.7 Other Standards and Specifications 66
5.2.7.1 AICC Guidelines and Recommendations (AGRs) 66
6. Glossary of Terms 68
7. References 70
Appendix B. Mobile Code Technology 73
Table B-1 Examples of Mobile Code Technologies 73
Appendix C. Configuration of the Integrated Learning Environment (ILE) 74
Table C-1 Configuration of the Integrated Learning Environment 74
Appendix D. Selected Standards 76
Appendix E. Brief Overview of a Game Engine Architecture 77

1. Introduction

The Navy is preparing to make significant investments in personal computer (PC) simulation-based learning systems. Presenting learning material in various formats and taking advantage of multimedia technologies has been shown to greatly benefit the learner. Simulation is one such technology that can provide enhanced learning experiences.

The purpose of this document is to help establish a foundation for investments in PC simulation-based learning systems and build consensus among those involved in their development and delivery. The document outlines the Navy's goals and objectives with respect to these systems. It is specifically focused on simulation-based learning systems that can be delivered over the Internet to common PC configurations. It does not address high-end simulations such as flight simulators or total immersion simulators that require special purpose hardware, facilities, and support staff.

The target audience for this document includes Navy Education and Training Command (NETC) claimants, Navy Systems Commands, and training-product development contractors. The document is intended to provoke thought and discussion. It may be consulted as a stand-alone reference on PC simulation-based learning or as a section in the larger Integrated Learning Environment (ILE) specification. Comments or corrections on the topics addressed herein are appreciated.

This document is intended to:

- Provide background on the motivation for PC simulation-based learning
- Set goals and objectives for PC simulation as a component of the Navy's Integrated Learning Environment (ILE)
- Define terminology for these systems that will be used in Navy acquisitions
- Propose a categorization scheme for PC simulation-based learning applications
- Suggest general criteria for determining the suitability or appropriateness of using simulation to implement learning applications that meet specific training objectives
- Specify a preliminary architecture, interfaces, and a common operating environment (COE) for the development, delivery, and usage of these systems
- Identify instructions, guidelines, and standards that are relevant to the development of PC simulation-based learning packages

It is not intended to:

- Present the science of learning or the latest learning theories of the research community
- Enumerate the specific training domains or applications where simulation-based training may be used effectively
- Provide detailed definitive or absolute criteria regarding the applicability of simulation to a particular learning application

- Address return-on-investment (ROI) calculations or provide detailed cost analysis criteria for specific training applications
- Detail general topics on learning systems that are otherwise addressed elsewhere in the Integrated Learning Environment (ILE) documentation

The remainder of Section 1 briefly describes why simulation-based training is important to the Navy, references studies and policies that are relevant to the application of this technology, and summarizes goals and objectives that are driving this effort. Section 2 introduces terminology that is relevant to PC simulation-based training and a categorization scheme for PC simulation-based training applications. It also suggests some general criteria for determining when this technology might best be applied to satisfy a specific training need. Section 3 presents an overview of system requirements, a preliminary architecture, and operating scenario for PC simulation-based learning systems. Section 4 provides an overview of standards organizations and associated standards specifications that are relevant to the development of these systems. The appendices provide additional supporting information, such as current recommended system configuration for target PC platforms, background on current Navy systems and tools, and more in-depth discussions of selected technical topics.

1.1 Why is PC simulation-based learning important to the Navy?

Simulation technology has been used in training systems within the Navy for many years. Simulation-based systems have been used to train aviation, surface ship, submarine, and shore support personnel alike. The systems have almost always been based upon custom hardware and software. Special purpose simulation-based training facilities have almost always been shore-based. These facilities usually require considerable resources to develop and maintain. Significant staff resources are needed to operate and support these simulators. Due to constant change and evolution of Navy systems, these custom facilities often have a limited useful life. Furthermore, trainees must travel and spend time away from their regular duty stations to take courses at simulation-based training facilities.

Research has shown that simulation has the capability of providing a more realistic and lasting training experience than some of the more traditional training methods that are often used, e.g., classroom lectures and traditional computer-aided instruction (CAI) methods. With the advent of the PC and the Internet, it is now possible to develop and deliver high quality simulation-based learning experiences at remote locations and satisfy many of the Navy's training objectives. A common operating environment based upon a minimal PC configuration with access to the Internet can provide widespread access to simulation-based training materials. These materials may be used in conjunction with classroom training, as part of operational duties, or as after-hours training.

PC simulation-based training applications can:

- Significantly reduce training time and can mimic real-time or non-real-time events
- Better mimic the actual work environment, thus producing better retention of learned skills
- Provide a *safe* environment for exercising 'what-if' scenarios or learning from one's mistakes

- Allow learning to take place without the need for using expensive operational equipment, thus reducing life-cycle costs
- Allow the safe practice of hazardous procedures

Simulation techniques can provide a robust and engaging learning experience, however, we must be cautious not to apply the technology for technology's sake. Better said, PC simulation will not meet all of the Navy's training needs, thus this technology should only be applied when there are valid reasons for doing so.

1.2 Relevant Studies and Policies

A specific finding from a Naval Operations (OPNAV) funded and Commander, Fleet Forces Command (CFFC)-directed analysis completed in April 2002 was a need to use PC-based simulation to support training objectives that don't require full mission trainers, at much lower cost, and are more likely to be deployable for use to refresh perishable skills. An underlying theme in the report was that schoolhouse simulations could be re-used to support individual training during pre-deployment work-ups if they were configured like the Fleet systems. PC-based simulations that are re-configurable and deployable on IT-21 compatible microcomputers aboard ship would meet the Fleet requirement and enhance relevance, quality and timeliness of training. The Naval Education and Training Command's Baseline Assessment Memorandum (BAM) 04 projected savings and a cost avoidance of $59M in Technical Training Equipment (TTE) maintenance over the Future Year Defense Programs (FYDP). PC-Simulation will be employed to reduce or offset TTE maintenance costs.

If PC simulation techniques can provide a more robust and engaging learning experience, why don't we make more frequent use of these technologies? PC simulations are often expensive to develop. Focus groups chartered by Brandon-Hall have estimated that typical development time for an e-learning course is around 220 hours of development for every hour of instruction. The development time required for a PC simulation is estimated to be at a ratio of 750-1300:1. Based on development time alone, an *immediate* need for a training product may preclude the use of PC simulation technology. Furthermore, equipment, process, or situational training requirements that change frequently may not be good candidates for PC simulation implementations.

The Navy Marine Corps Intranet (NMCI) is a long-term initiative between the Department of the Navy (DoN) and the private sector to deliver a single integrated and coherent department-wide network for Navy and Marine Corps shore commands. Under NMCI, EDS is responsible for providing all IT hardware and software, operations, training, maintenance and system upgrades. This ensures standardization and interoperability. Currently each command addresses their information technology requirements on an individual basis. EDS and their partners will provide comprehensive, end-to-end information services for data, video and voice communications for DoN military and civilian personnel and deliver global connectivity to make our workforce more efficient, more productive, and better able to support the critical war fighting missions of the Navy and Marine Corps. NMCI will provide significantly greater security than our heterogeneous ashore networks have today. These networks are vulnerable to security threats, ranging from hackers to denial attempts, because security implementation varies greatly by location and the many components responsible for system certification and authentication. NMCI provides one centrally managed secure network linking all shore commands using the defense-

in-depth security architecture and encrypted authentication mechanisms. For more on information security policy, see (DODJS 2003). Additionally, the NMCI contract provides incentives for the vendor's performance in secure operation of networks. The NMCI Gold Disk provides a common set of software applications including operating system, office environment, web browser, file compression tools, security software, etc., see http://www.nmci-isf.com/downloads/Gold_disk_contents.pdf.

PC simulation products that are intended to run on the NMCI environment must get proper approval and be subjected to appropriate testing. NMCI Application Ruleset Version 2.96-2 July 200, provides guidance concerning any software that generates or allows the user to create programming code which compiles into executable (.exe) files that are installed and can be run from the user's workstation. For more information, see http://cno-n6.hq.navy.mil/navcio/file/NMCI-RuleSet-2_96-2Jul03.doc and the Navy Enterprise Application Development Guidelines, Supplement A.

What is Information Technology for the 21st Century (IT-21)? "Recognizing the need for a system that would allow rapid dissemination of information between all U.S. forces, the DoD earmarked a PC-based tactical and support war fighting network known as IT-21. The goal of IT-21 is to link all U.S. forces together in a network that enables voice, video and data transmissions from a single desktop PC. In short, IT-21 sets forth minimum standards for the procurement of computer software." For further information on IT-21 and the development of training materials, see (NPDC 2003).

Contractors are not to incorporate in contract deliverables copyrighted material that is not first produced under the contract without either granting to or acquiring for the Government certain copyright license rights for the data or obtaining permission from the contracting officer to do otherwise. See Federal Acquisition Regulation ("FAR") FAR 27.104(h) and 27.404(f); DFARS 252.227-7013(d) and http://www.paxriver.org/files/PropertyRights.doc.

1.3 Goals and Objectives

The goal of the PC simulation-based learning effort is to improve fleet readiness and provide enhanced training experiences through the appropriate development, effective use, and re-use of simulation technology. Objectives supporting these goals fall into three major areas: use of common computing platforms and software, guidelines for developers, and management and delivery infrastructure. Although many of these objectives are shared with those of the overall Integrated Learning Environment (ILE), in many cases they have special significance for PC simulation-based learning. This is due in part to the fact that these objectives in many respects are pushing the current technology. The current objectives are:

Use of Common Platforms

- use of a common low cost PC hardware platform
- specify common software resources, e.g., operating systems, browsers, plug-ins
- ensure that common platform and software configuration provides suitable performance with respect to training materials downloads and simulation package execution

Guidelines for Developers

- enable the use of latest proven multimedia, simulation, and learning technologies
- where possible, avoid building to lowest common denominator
- provide for reuse and sharing of course content objects to maximize Navy investment and minimize re-development costs through the use of appropriate industry and Navy standards
- identify relevant standards, development tools, and practices, including: object coding, level of object granularity, and tagging schemes for sharable content
- provide for migration of existing course content to new technology as required

Course Management and Delivery Infrastructure

- provide a centralized repository for managing and delivering course materials
- establish procedures for the submission of new course materials and software updates by course content providers
- provide testing mechanisms to ensure compatibility of training materials with common operating environment
- establish procedures for the download and installation of training materials
- provide for centralized tracking of student progress and package usage
- provide mechanisms for problem/bug reporting
- support centralized management and notification of software updates
- establish access control and security mechanisms
- manage licensing and intellectual property copyright issues

2. Terminology, Categorization, and Criteria

This section defines key terminology associated with PC simulation-based training (Section 2.1), outlines a categorization scheme for simulation-based learning applications (Section 2.2), and suggests general criteria for determining the suitability or appropriateness of simulation technology for different types of learning applications (Section 2.3).

2.1 PC Simulation-based Learning Terminology

This section introduces terminology pertaining to instructional simulations that are designed to run on general purpose PCs using common operating systems and peripheral devices. PC simulation-based courses may be taken inside or outside of classroom environments. Training materials are delivered electronically from centralized distribution facilities to the student's computer. Depending upon security constraints, installation of course materials and supporting software may be performed either by the student or Navy computer support staff.

The terminology that is addressed in this section falls into the following categories:

- Instruction and learning modes
- Content management and delivery systems
- Simulation model types
- Simulation system elements
- Quality of the simulation experience

A brief discussion of each category follows.

2.1.1 Instruction and Learning Modes

Instructors may use PC simulation-based learning systems to supplement classroom instruction at learning centers. Alternatively, simulations may be integrated into computer-based training courses that are independently accessed by students (i.e., outside of a classroom environment) to pursue their own professional development objectives or obtain on-line help. The associated instruction and learning modes are defined as:

- ***Electronic Classroom Synchronous (ECS)*** – This learning mode is based on instructor-led training. It the model for Navy Advanced Electronic Classrooms (AEC). Instructors ideally should be able to preset programmable system faults the student is expected to diagnose, and control various, features, and object parameters to tailor the lesson to individual students and take advantage of other AEC functionality.
- ***Electronic Classroom Asynchronous (ECA)* –** This mode is based on an instructor facilitator, as opposed to instructor-led training. Navy Advanced Electronic Classrooms (AEC), Navy Knowledge Online (http://www.nko.navy.mil/), and Learning Resource Centers (LRC) support this model of instruction.
- ***Formal Self-Guided Instruction* –** An alternative learning mode is based on courseware that allows the student to have ultimate control over self-paced lessons that are taken outside of a classroom environment. This type of training technology is also known as Interactive

Courseware (ICW) or Interactive Multimedia Instruction (IMI). An advantage of this mode is that the learner can make use of the instructional material virtually anytime, anywhere.

- ***Informal On-the-Job Tutorials –*** With the advent of laptop computers and portable digital assistants (PDAs), ad hoc simulation-based training experiences are possible on the job as an extension of on-line help capabilities. In this learning mode, the individual has a simulation-based learning experience in the process of performing a normal job assignment, such as system operation or maintenance. As a result of accessing an on-line "help" system via a laptop or PDA, a simulation-based course content object is activated to allow the individual to learn more about the operation or maintenance of the system.

2.1.2 Content Management, Delivery, and Tracking Systems

Traditional course content distribution schemes have been based on the management and transfer of physical computer media, e.g., CD-ROMs or floppy diskettes. These schemes are costly, time-consuming, and difficult to manage. The media must be managed, i.e., physically stored and tracked. The media has to be physically generated, delivered, and installed on each user's computer. Lost or damaged media must be replaced. Delivery to distant locations delays student access to courses. Each time course content is updated, the entire process must be repeated. Furthermore, this mode of operation has not provided efficient mechanisms for tracking and reporting student, progress, scores, completion, and summary statistics across many users.

Electronic delivery and tracking of courses over the Internet, local area networks, satellite links, and/or phone lines could significantly reduce these problems. For these reasons, the Navy will electronically manage and deliver PC simulation-based learning content wherever possible. Student progress will also be tracked by electronic means. The Navy's *Integrated Learning Environment (ILE)* has been established to address these issues. A general understanding of the ILE is important to courseware developers, Navy IT managers, and users of PC simulation-based training materials. Major components of the ILE include:

- **Learning Content Management System (LCMS)** – Functions as a repository for course content objects and as a courseware authoring system. Courseware developers and Learning Management Systems will retrieve course materials and course content objects from the LCMS. Course developers may use authoring systems other than the LCMS to develop course content objects, but ultimately all objects must be stored in the formats and delivered via the protocols of the ILE. Courseware deliverables, i.e., course content objects, will be made available through the LCMS after they have been tested to ensure that they function properly and are in compliance with appropriate standards.
- **Learning Management System (LMS)** – Functions of the Learning Management System are similar to those associated with the administration of the academic program at a university. It is responsible for publishing a catalog of course offerings, providing mechanisms for course registration and checking of prerequisites, maintaining transcripts for the student population, and tracking student progress, scores, completion, etc. Different Navy organizations are currently using different Learning Management Systems. The systems are run in different physical locations.
- **Course Management System (CMS) –** Functions of the CMS include student testing, remediation, and tracking of the start and finish of various modules ***within*** a training course. A CMS is also known as Computer-Aided Instruction (CAI) or a Course Management Shell.

The CMS typically provides collaboration tools such as asynchronous discussion forums, audio/video chat, text chat, email facilities. These capabilities are needed for courses that are conducted more effectively with instructor facilitation and student interaction. Most CMS products have LMS functions. They may or may not be compatible with a specific LMS implementation. Courseware must include minimal CMS functions to exchange information with the LMS such as bookmarks, start date/time, completion date/time, grades, etc.

For more information on these systems, see the ILE specification on the Navy Knowledge On Line web site (http://www.nko.navy.mil/), or the architecture discussion in Section 3.2 of this document.

2.1.3 Simulation Model Types

In The Handbook of Simulation, Jerry Banks defines simulation as:

> *"…the imitation of the operation of a real-world process or system over time. Simulation involves the generation of an artificial history of the system and the observation of that artificial history to draw inferences concerning the operational characteristics of the real system that is represented. Simulation is an indispensable problem-solving methodology for the solution of many real-world problems. Simulation is used to describe and analyze the behavior of a system, ask what-if questions about the real system, and aid in the design of real systems. Both existing and conceptual systems can be modeled with simulation*." (Banks, 1998)

The Department of Defense (DOD) defines a simulation as a model that represents activities and interactions over time. A simulation may be fully automated (i.e., it executes without human intervention), or it may be interactive or interruptible (i.e., the user may intervene during execution). A simulation is an operating representation of selected features of real-world or hypothetical events and processes. It is conducted in accordance with known or assumed procedures and data, and with the aid of methods and equipment ranging from the simplest to the most sophisticated.

A simulation model may be defined as a representation of some or all of the properties of a device, system, or object. There are three basic classes of models: mathematical, physical, and procedural.

- **Mathematical model** – a representation comprised of procedures (algorithms) and mathematical equations. These models consist of a series of mathematical equations or relationships that can be discreetly solved. Usually the models employ numerical approximation techniques to solve complex mathematical functions for which specific values cannot be derived (i.e., integral). The basic Anti-Submarine Warfare training program uses a computer generated animation to illustrate the principles affecting sound propagation in water. Students are able to see visual representations of complex mathematical models.
- **Physical model** – a representation of the physical object or system and its relationship to other real world objects. For example, an operating model of the components of a pump or an engine.

- **Procedural model** – an expression of dynamic relationships of a situation expressed by mathematical and logical processes. Joint warfare operations, tactical team exercises, and advanced operator training such as aircraft controller, and air intercept operations may benefit from procedural models.

A PC simulation is a desktop or laptop computer software program that strives to mimic a phenomenon, experience, equipment or environment that is based on reality. Fantasy games are purposely excluded. When applied to a training domain, the simulation serves to provide the user with the opportunity for learning in a robust, motivating, and engaging environment. The presentation of the material is optimized by a high degree of user interactivity, fidelity and immersion, and where context and practice are the key to learning (see Section 2.1.5).

A simulation is **not** in itself a "canned" animation such as an AVI or MPEG file. With these animation files, there is no interaction with the trainee – the outcome of the animation is predetermined and unchanging. However, it is possible that canned animations might be used as a part of simulation experience or may be used to introduce a simulation in larger training packages.

2.1.4 Simulation System Elements

The approach that the Navy is taking to PC simulation-based learning brings together elements from several different technology areas. The technology areas include web browsers, simulation engines, distributed simulation, and simulation-based learning. Terminology from each of these areas is briefly described.

2.1.4.1 Browser Elements

The functioning of the World Wide Web is based upon standards for communicating, displaying and linking web pages that contain text, graphics, and multimedia resources. These standards enable worldwide access to web pages from disparate types of computers through the use of *web browsers*, such as Microsoft Internet Explorer and Netscape Navigator. *Web browsers* have common navigation mechanisms, i.e., point and click controls, that allow users to move around web sites, view or save information, and perform other functions. *Browsers* transfer, or "download," requested web pages from a web server to the requesting client computer's memory and display the page's contents. With this functionality and multi-media resources alone, fairly sophisticated learning applications can be constructed.

Simulation-based learning applications will typically require more powerful functionality than traditional web page downloads. Simulation involves user interaction with a dynamic executable model. Simulations could conceivably run on either a server or a local client computer. The problem with running simulations on a remote server is communications bandwidth and fidelity. If the simulation involves the generation of high fidelity visual imagery or audio, the communications bandwidth afforded by the Internet may be insufficient to support the quality of simulation experience required. For more on the quality of the simulation experience, see Section 2.1.5.

Mobile code and *browser plug-ins* can be used to create executable simulations on client computers. *Browsers* can download *mobile code*. *Mobile code* is defined as software obtained

from remote systems, transferred across a network, loaded, and executed on a client computer system without explicit installation or execution by the recipient. *Mobile code* has the ability to migrate from one computer to another to extend the program's capabilities. For example, a simulation program could move from a server to a client computer. This would be done when the transfer of data generated by the simulation would require too much network bandwidth between the server and the client. The downside of mobile code is that it may create security problems. The challenge is to ensure that downloaded code does not compromise the security of the client system. A list of example mobile code technologies is provided in Appendix B.

A *browser plug-in* is an extension of the browser's capabilities. A number of multimedia players have been implemented as *plug-ins*. Although *browser plug-ins* can be downloaded over a network, they are not mobile code, per the previous definition, because they require approval at the client location to install. A *plug-in* could conceivably access and take advantage of any hardware, operating system, or software application capability on the client computer. *Plug-ins* could be used to construct virtually any kind of simulation functionality that would be required on a PC. *Browser plug-ins* are also a potential security problem for this same reason. As such, the Navy has established policies regarding the testing and installation of plug-ins, see Appendix A.

2.1.4.2 Simulation Engine Elements

The construction of a simulation usually involves some sort of software development. Since software development is often a costly, time-consuming, and error-prone process, minimization of programming and re-use of validated code is highly desirable. A certain amount of code re-use can be achieved through the utilization of *simulation engines*. These *engines*, or *simulators*, are computer programs that typically provide functions to:

- Develop and manage simulation models
- Implement control logic and perform calculations
- Assist in model debugging
- Incorporate programming language extensions
- Input and output data
- Initiate and terminate simulation runs
- Generate statistical variations between runs
- Create and display 2D and/or 3D visualizations
- Analyze results
- Produce reports

By using a *simulation engine*, much low-level coding could be avoided. Unfortunately, most *simulation engines* currently run as stand-alone systems on personal computers. As such, new systems, or modifications to existing systems, will be required before *simulation engines* will be able to run as browser plug-ins, mobile code, or run remotely on a server and interact with browser plug-ins. This change in implementation philosophy will be critical to meeting the Navy's requirements for electronic distribution and management of PC simulation-based courseware.

The representation of time is a key aspect of *simulation engines.* They typically implement models as discrete-change, continuous-change, or combined-change, see (Banks 1998). In discrete-change *simulation engines*, data variables only change values at distinct points in simulated time, i.e., discrete events. In *engines* that implement continuous models, data variables may change values continuously as functions of time. With combined models, variables may change discretely, continuously, or both. Some of the common elements found in *simulation engines* include a clock (that keeps simulated time) and system state variables (that indicate the current state of the simulation execution). Other elements include representations for entities (the objects whose behavior is simulated), events (points in time where things happen), event queues (a sorted list of events), and random number generators (that are used to create variations in the simulation executions).

Some of the most common techniques for implementing and extending models using *simulation engines* include the use of conventional procedural code, object-oriented programming, rule-based systems, and/or finite state machines. A discussion of these techniques is beyond the scope of this document, for more on this and related topics see software engineering texts such as (Sommerville, 2000).

2.1.4.3 Distributed Simulation Elements

Although PC simulation-based learning systems will often be implemented as individually-executable computer processes, sometimes there will be a need to divide simulations into multiple processes. These processes may need to run as a distributed simulation system on a single PC or over a network on multiple PCs. A distributed simulation system may be used to:

- Divide a large simulation into smaller functional modules that can be used by multiple training packages
- Provide a simulation service to a learning system client
- Enable coordinated simulations for multi-student exercises over a local classroom Intranet or the Internet.

The *High Level Architecture (HLA)* is a standard, originally initiated by the DoD, for implementing distributed simulation. In *HLA* terms, the individual simulations are called *federates* and the distributed simulation is referred to as a *federation*. The *HLA* defines a framework by which individually executing *federates* can be combined into a distributed simulation *federation*.

The *HLA* framework has three major parts. The first part is a set of rules that *federates* and *federations* must adhere to ensure that a federation operates properly. The second part is a software system called the *Run Time Infrastructure (RTI).* The *RTI* defines an interface that provides a number of services that *federates* can use to communicate (i.e., exchange simulation data), and coordinate their execution (i.e., synchronize simulation clocks) with other *federates* in a *federation*. The third part of the *HLA* is called the *Object Model Template (OMT).* The *OMT* provides a means for describing the format of the data that will be exchanged between *federates*. For more information on distributed simulation using *HLA*, see Section 4.2 of this document and (Kuhl, Weatherly, and Dahmann, 1999).

2.1.4.4 Simulation-based Learning Elements

PC simulation-based learning applications may require that the student interact in a simulated environment, that is, a virtual world. There are several terms that are useful in describing virtual worlds. A *world model* is often used to describe the internal representation of the environment that is being simulated, even if it is fairly simple, such as the model of a pump. Inside the world model, objects that perform dynamic actions (i.e., programmable objects) are commonly called "*actors*" or "*agents*." This term applies to objects that one would normally think of as being inanimate, as long as they have associated actions. For example, a block of ice might be implemented as an actor because its action would be to change state, i.e., melts, as the surrounding temperature rose above freezing.

Inside the virtual world, the representation of the user, whether it is a simple icon or an actor, is commonly referred to as an *avatar*. More sophisticated representations of *avatars* and other actors (e.g., people and machines), may be given the ability to carry out complex actions using the simulator programming extensions identified in Section 2.1.4.2. The terms that are often used to describe these complex actions are *primitive motions*, *skills*, and *behaviors*. A *primitive motion* may simply involve rotating a character's wrist joint. A *skill* may involve a simple sequence of primitive motions, such as reaching for and grasping a tool. A *behavior* may represent a much more complex set of *skills*, such a removing a spark plug from an engine. This *behavior* might require the execution of a sequence of *skills*: 1) removal of the spark plug cable, 2) tool acquisition and setup, and 3) plug removal with the tool. Programming methods that may be used to implement *behaviors* include procedural code, object-oriented code, hierarchical finite-state machines, and rule-based systems. Textbooks on artificial intelligence provide useful introductions to this technology; see (Winston, 1992).

Simulation-based learning will typically involve *role-playing* activities. An example of a *role-playing* activity may be the maintenance of a piece of Navy equipment. Scenarios and definition of student *roles* are important to *role-playing* simulations, e.g., a piece of equipment is broken - the student must diagnose and correct the problem. The scenario defines the goals and the simulated world in which the goals must be pursued. The *roles* define the way the student's *avatar* will behave in the simulated world.

Scaffolding is a term used in simulation-based learning to refer to structures built into the scenario, simulated world, roles, and tasks that help the student to achieve the desired learning experience. The *scaffolding* aids the student in achieving an outcome that would be difficult or impossible without support.

2.1.5 Quality of the Simulation Experience

Simulations are designed to take the place of real first-hand experience. In many cases, real first-hand experience may be dangerous or costly to reproduce in a training environment. So, how does the simulated experience compare with the real experience? Fidelity, interactivity, and immersion are three terms used to describe the quality, i.e., richness of the simulation experience. Each of these terms is introduced below.

2.1.5.1 Fidelity

Fidelity is the accuracy of PC simulation object representation when compared to the real world. Generally, the higher the fidelity required for the training product, the higher the development costs. Within a PC simulation, fidelity will be a measurement of how accurately a real-world sound or visual image is reproduced. Audio and visual fidelity is described in more detail below.

Audio Fidelity - Audio fidelity is determined by the sampling rate, i.e., the quality of the original audio and resultant presentation. It is a function of:

- Input Sampling Rate
- Compression/Decompression (CODEC) techniques
- Mono or Stereo delivery

Increasing levels of audio fidelity impact the PC simulation delivery options (due to throughput and bandwidth considerations) and could substantially increase the cost for development. For example, real-time audio for situational simulations typically needs a higher sampling rate and needs to be cognizant of bandwidth limitations as opposed to recorded voice audio. PC simulation training course procurers and developers need to select the level of audio fidelity appropriate to meet the training requirement.

An 8 KHz sampling rate is sufficient, but not always ideal for voice and equipment sound mimics. Lower fidelity often is presented in mono vice stereo. On the high-end of the spectrum, CD-quality music is sampled at 44 KHz. These digital audio files can get quite large. For example, if we were to store one minute of CD-quality audio, we would have 44,100 samples per second or 2,646 samples at 16 bits each for a total of 5.2 megabytes of data per channel. This is over 10 megabytes of data for 1 minute of CD-audio quality of music. To save disk space and bandwidth, we can reduce the sampling size or the frequency of sampling, but the *fidelity* on playback would suffer in terms of introducing audible hiss, noise, or abrupt changes in sound.

Visual Fidelity - Visual or image fidelity is a function of:

- Color depth
- Size (physical size and resolution)
- Shading fills and light focus
- Frame rate …(motion)
- Depicted detail

Tradeoffs can be made for visual or image fidelity for considerations of delivery options, required throughput, or bandwidth limitations so long as it does not detract from the intended learning objectives. The higher the resolution required -- the greater the amount of data generated. The trick is to end up with a significantly smaller amount of data with no perceptible loss in *fidelity*. PC simulations can make use of various video compression techniques to meet the minimal training objectives.

2.1.5.2 Interactivity

For the purpose of categorizing PC simulation, interactivity has several forms:

- Hierarchical interactivity
- Support interactivity
- Object/Construct interactivity
- Simulation control interactivity
- Contextual interactivity

Each of these types of interactivity is described in more detail below.

Hierarchical Interactivity - Varying levels of hierarchical interactivity provide the learner the ability to navigate through the domain-knowledge representations. This form of interactivity gives the learner control in exploring the subject in a self-driven mode. The associated hierarchical interactivity mechanisms could be menus, hyper-links, navigational bars, structured listings, XYZ-coordinates, value setting/scroll bars, axial rotation, zooming, etc.

Support Interactivity - Support interactivity describes the afforded feedback mechanisms and performance support to learner reactive inquiry (context sensitive and insensitive). The complexity of support interactivity could range from a context-insensitive or context-sensitive reference system (e.g., help files).

Object/Construct Interactivity - Object interactivity affords the learner a means of proactive inquiry. This type of interactivity engages the learner in manipulation of real world representations of the objects (buttons, dials, radial boxes, people, things etc.) that are activated by an input device such as a mouse. Construct interactivity requires the creation of an instructional environment in which the learner is required to manipulate component objects to achieve specific goals and contains a feedback as a result to a learner response to a generated problem. Generally, the more complex the modeled object behaviors and faults, and the more number of objects modeled, the more complex and expensive the training product development activity becomes.

Simulation Control Interactivity - Simulation interactivity is the ability to simulate several aspects of the real world in a realistic and highly representative way. The learner or instructor should be able to select what aspects to include in, or exclude from, the simulation and set the modeling parameters and characteristics. The simulation's robustness and real-time response are crucial elements of this type of interactivity.

Contextual Interactivity - This concept combines and extends the various interactive levels into a complete virtual training environment in which the learner is able to work in a meaningful, job-related context.

2.1.5.3 Immersion

Immersion can be defined as complete attention, intense mental effort, absorbed, or engrossed. To engage the learner, there is a need for “learning by doing.” The more the PC simulations engage the learner, the higher the level of immersion applied to the intended learning objectives. The level of immersion for PC simulations can additionally be tied to both interactivity and fidelity. The following levels of immersion are defined for the Navy PC simulation domain:

- ***Desktop Immersion*** **-** Besides the monitor used to display images stressing the portrayal of job context and audio augmentation, no other sensorial output is used. Traditional computer applications could be put in this category.

- ***Limited Multi-Sensory Output Devices*** **-** Incorporates specialized displays, earphones, and tactile or force feedback devices only when critical for meeting specific requirements of the training objectives. Note: Not all NMCI workstations have speakers. Any mission rehearsals or team dynamics training requiring collaboration or role-playing among the participants that is augmented and coordinated with audio/visual tools are included in this level of immersion.

The PC simulation domain does **not** include providing the illusion of being inside the simulated environment, i.e., levels of presence. Nor does PC simulation include 'fish tank' immersion that is typically achieved by providing stereoscopic images and tracking the user's head or direction of sight in order to simulate a motion parallax effect.

2.2. Categorization Scheme of Simulation Types

Five simulation types are proposed as relevant to Navy PC training needs. The types were derived and adapted from the previously mentioned Brandon Hall study. The PC simulation types are:

- Cognitive support
- PC software
- Situational
- Procedural
- Virtual world

Each of these simulation types is briefly described below.

2.2.1 Cognitive Support Simulation

Based on the principle of "a picture is worth a thousand words," *cognitive support simulation* presents models of complex concepts to assist the mental process or faculty of knowing. These mental processes include such aspects as awareness, perception, reasoning, language, memory and judgment. These simulations can range from simple dynamic diagrammatic representations to complex three-dimensional models. The goal of *cognitive support simulation* is to assist learners in the acquisition, organization, and application of knowledge. These simulations may rely upon motion to explain complex concepts. They are often simple to construct and lend themselves to re-use in various training products.

2.2.2 PC Software Simulation

PC software simulation can be used to demonstrate step-by-step instructions for using a PC software application. This model can be extended to other non-PC workstations and custom hardware through screen captures. Software demonstration and prototyping packages are available that allow the developer to incorporate dynamic interactions into imported screen captures.

This type of simulation may be used to demonstrate and teach software applications in other hardware and operating system environments (e.g., Unix-based workstations). Example training applications might include *PC software simulations* of Command, Control, Communications, and Computer (C4) Systems, Global Command and Control Systems (GCCS), and Theater Battle Management Systems (TBMS). The development of *PC software simulations* is for the most part simple and inexpensive. These simulations can range from simple demonstrations of application interactions intended for passive observation to highly interactive lessons that engage the learner to practice their PC software skills. By combining *PC software simulation* with *procedural simulation*, refresher system and/or equipment operator training could be provided.

2.2.3 Situational Simulation

Situational simulations are typically developed to assist learners in problem solving, soft skills and team dynamics training. They are usually based on role-playing simulations with case-based scenarios. *Situational simulations* are ideal for mission rehearsal. In these simulations, learners are typically members of the environment, rather than an external force that manipulates variables at will. They often incorporate situations in which participants react to many decision alternatives and feature a best—or optimal—sequence of decisions, see Brandon-Hall, 2003.

Situational simulations will likely use complex flowcharts or state tables to map out the desired role-playing scenarios. When using *Situational Simulation* for team coordination and collaboration training wherein human interaction is critical to mission rehearsal, augmentation with audio/video collaboration tools would be desirable.

2.2.4 Procedural Simulation

Procedural simulations are ideal for systems and/or equipment operation and maintenance training. Start-up procedures, task sequences, and other practice drills are ideal candidates for *procedural simulation*. *Procedural simulations* afford a safe, realistic environment that resembles, as close as possible, the actual experience of operating and/or maintaining systems and equipment. They allow the learner to immediately see the results of their actions, i.e., learning by making mistakes. These types of simulations are potential candidates for replacing Technical Training Equipment (TTE) in Navy Schoolhouses.

These simulations can be designed to use very advanced state tables, variable tracking and triggers that change the states of the modeled system/equipment as the learner performs various activities. The cost of developing a *procedural simulation* is directly proportional to the level of fidelity, interactivity and student/instructor control provided. Fidelity and interactivity are manifested in the number of objects, system/equipment functions, events, pre-programmed faults, and feedback mechanisms that are modeled. See Section 2.1.2 for discussions on fidelity and interactivity issues.

2.2.5 Virtual World Simulation

A *virtual world simulation* often involves the navigation of a synthetic space. Using a photo-realistic computer-generated interactive environment, these simulations strive to achieve the same sense of space as in the real world through motion and visual cues. Subjective measures based on human spatial perception supplementary to accurate geometry, illumination, and task

performance, reveal the actual cognitive mechanisms in the perception of a virtual environment that are not otherwise apparent.

2.3 Suggested Simulation Suitability Criteria

PC simulation-based learning can benefit any type of training wherein practice is required to become proficient at a skill. The more critical it is for the practice to take place in context, the more PC simulation will benefit the training type.

2.3.1 Learning Levels

Table 2-1 provides a synopsis of learning levels that may benefit from PC simulations. For a more detailed discussion on learning levels, see [Bloom, 1964]. The table categorizes learning levels within the three learning types. It associates action verbs with their definitions for each of the learning levels based on the psychology of learning. Each of the action verbs can be compared with job task lists and training/learning objectives for an identified training need. Using this approach, it becomes clear which learning levels could be considered appropriate for the use of PC simulation products.

Table 2-1 Sample Learning Levels for PC Simulation Consideration

Learning Type	Learning Level	Description
Knowledge		The fact or condition of knowing something with familiarity gained through experience or association
	Discrimination Learning	Learning to group similar and dissimilar items according to their distinct characteristics.
	Fact Learning	The learning of verbal or symbolic information (e.g. names, formulas, facts)
	Problem Solving	Learning to synthesize lower levels of knowledge for the resolution of problems.
	Procedure learning	Learning to perform step-by-step actions in the proper sequence.
	Rule Learning	Learning to use two or more facts in a manner that will provide regularity of behavior in an infinite variation of situations.
Skills		The ability to use one's knowledge effectively and readily in execution or performance. Dexterity or coordination especially in the execution of learned physical tasks.
	Adaptation	Learning to modify a complex physical skill to accommodate a new situation
	Continuous Movement	Learning to track, make compensatory movements based on feedback.
	Perception	Perception of sensory stimuli that translates into physical performance.
	Readiness	Learning to have Readiness to take a particular action.
Attitude		A mental position with regard to a fact or state. A feeling or emotion toward a fact or state
	Competence	Learning and demonstrating the mental preparedness to make decisions by using prioritized strategies and tactics in response to normal, abnormal, and emergency condition cues associated with the performance of an operational procedure.
	Receiving	Learning and demonstrating the ability to perceive the normal, abnormal, and emergency condition cues associated with the performance of an operational procedure. Situational Awareness of operational condition cues.
	Responding	Learning and demonstrating mental preparedness to encode operational cues as indicators of normal, abnormal, and emergency conditions associated with the performance of an operational procedure.
	Valuing	Learning and demonstrating the ability to judge the worth or quality of normal, abnormal, and emergency cues associated with the performance of an operational procedure.

2.3.2 Training Applications and Simulation Types

What types of learning applications are appropriate to each of the categories of simulation? Table 2-2 provides examples of different training applications and the simulation types that may be used according the categorization scheme introduced in Section 2.2. A complete enumeration of Navy PC simulation-based learning applications is beyond the scope of this document.

Table 2-2 Types of Training Benefited by PC Simulation

Cognitive Support Simulations	
Complex Concepts training	Familiarization Training
Applied Theory Training	
PC Software Simulations	
PC Application Tutorials	Tactical Workstation Application Tutorials
System Operator Training	Electronic Performance Support Systems
Situational Simulations	
Just-In-Time (JIT) Refresher training	Role-playing
Team Dynamics training	Collaboration training
Mission Rehearsal	Interpersonal Skills, Soft-Skills Training
What-if Scenarios	Scenario/Case-based Training
Decision-making Skills Training	
Procedural Simulations	
System/Equipment Operator Training	System/Equipment Maintenance Training
Just-In-Time (JIT) Refresher Training	Mission Rehearsal
Job Training Aids	What-if Scenarios
System Calibration Training	Safety Training
Time-based Training	Sequence Training
Electronic Performance Support Systems	
Virtual Worlds	
Role-playing	Team Dynamics Training
Mission Rehearsal	Familiarization Training

2.3.3 Evaluation Methodology

PC Simulation could be used in specific types of training over other alternatives, such as live exercises or classroom lectures, because of potential cost savings, flexibility, and/or fidelity. Human Performance and Instructional Systems Design practitioners should make this determination. These experts have an in-depth knowledge of the “science of learning” and Navy’s four-quadrant model.

The Human Performance System Model (HPSM) is the cornerstone of the CNO's Revolution in Navy Training. Utilizing this dynamic four-quadrant model, Navy educators will know the defined outcome of training and continually measure and adapt the product. While training is fundamental to Navy readiness, the HPSM is the tool by which the Navy will refocus its training on the Sailor's professional and personal growth and development. The Sailor becomes the focal

point of the new education system, not the hardware, and the key to its success will be the HPSM. The four quadrants are:

- Step 1 - Define human performance requirements by breaking down jobs and job tasks into specific behaviors and determining the knowledge, skills and abilities Sailors need to do their job. Once defined, these requirements are validated and prioritized by CFFC (Commander, Fleet Forces Command), or other equivalent decision-makers.

- Step 2 - Consultants and subject matter experts analyze the requirements and recommend solutions containing the tools necessary to achieve the desired improvements in human performance.

- Step 3 - Solutions are then developed, built and integrated in the third step of the process. Solution options may include traditional classroom instruction; eLearning applications; simulations, models or games; on-the-job-training and more.

- Step 4 - solutions are put into practice and their outcomes are evaluated for their effectiveness in meeting the human performance requirements established in the first step. The results of the evaluations are then fed back into the first step and the process is changed or adapted accordingly. Thus, the cycle continues to rejuvenate the process and ensures the Navy's educational delivery system keeps pace with changing technologies and associated human performance requirements.

See (HPC 2003) and http://www.excel.navy.mil/human.htm for more on the Navy's four-quadrant model.

Instructional Systems Design Perspective involves determining if PC simulation is appropriate to meet the training objectives through a review of the learning levels within the three learning types (knowledge, skills, and attitudes). PC simulation training categories (Section 2.2) will additionally assist in this determination. To determine if PC simulation is a candidate technology to meet a training need, the follow steps are recommended:

1. Establish a need from the human performance and instructional systems design perspective (Section 2.3.1)
2. Determine the minimal level of interactivity, fidelity and immersion needed to meet the training need (Section 2.1.2).
3. Determine the appropriate simulation implementation mechanism to meet the training need, e.g., browser-based simulator, stand-alone simulator, or distributed simulation (See Sections 2.1.3 and 3.2).
4. Select the categories of PC simulation that are appropriate to the training need (Sections 2.2 and 2.3.2).
5. Conduct a Return on Investment (ROI) Analysis.

It is often difficult to determine whether simulation is justified to meet a specific training need due to its high cost of development. Calculation of the return on investment (ROI) is essential when deciding the appropriateness of PC simulation for training. Return of investment (ROI) is

commonly defined as the ratio of profit/benefit to the amount invested. ROI is usually calculated using dollar terms.

It is a non-trivial task to establish an aggregate value for PC simulation. Estimation of the value of PC simulation over other alternatives requires a cost and benefits analysis. The cost of development must be taken into consideration when conducting an ROI analysis. PC simulations should be procured with the minimum level of fidelity, level of interactivity, object parameter configuration (number of pre-programmed and configurable faults and events), and immerse realism to meet the training need.

Indirect, intangible benefits are sometimes difficult or impossible to quantify. Tangible benefits easily translate to cost quantities, but intangible benefits are seldom expressed in terms of dollars. In some situations, cost may not be the most important metric, intangible benefits can be very important. Intangible benefits may include avoiding the loss of life or tactical assets through the demonstration of hazardous procedures or learner engagement in drills and practice on mission critical procedures. Examples of tangible and intangible benefits to be considered when calculating ROI are given in Table 2.3.

Table 2-3 Benefit Considerations for ROI Calculations

Tangible Benefits	Intangible Benefits
• Productivity/throughput	• Efficiency
• Personnel cost	• Safety/health of personnel
• Equipment cost	• Readiness
• Maintenance cost	• Learning
• Travel expenses	• Motivational aspects
• Facility cost	• Customization
• Time	• Risk Mitigation
• Logistics (Life Cycle Support)	

Calculation of the ROI associated with a specific PC simulation-based learning application is beyond the scope of this document.

3. PC Simulation-Based Learning Architecture

What is architecture and why is it important to PC simulation-based learning systems? In Shaw and Garlan's book the significance of software architectures is explained:

> *"As the size and complexity of software systems increase, the design and specification of overall system structure become more significant issues than the choice of algorithms and data structures of computation. Structural issues include the organization of a system as a composition of components; global control structures; the protocols for communication, synchronization, and data access; the assignment of functionality to design elements; the composition of design elements; physical distribution; scaling and performance; dimensions of evolution; and selection among design alternatives. This is the software architecture level of design." (Shaw 1996)*

This section introduces a summary of requirements that are driving PC simulation-based learning in the context of the Navy's Integrated Learning Environment (ILE) and presents a preliminary architecture for PC simulation-based learning. These two topics are addressed generically. Mention of commercial products is avoided in this section. The specific commercial hardware and software systems that are used to fulfill requirements and implement modules can be found in the appendices to this document.

3.1 Requirements Summary

The Navy will be largely dependent upon learning systems that are commercially developed and supported. Simulation systems will be constructed through the integration of commercial off-the-shelf (COTS) software applications, simulation models, and course content objects that are obtained from different learning application developers and vendors. The COTS applications to be integrated include not only simulation systems, but also other software applications such as browser plug-ins that support visualization, etc. As such, the existing functions, capabilities, and interfaces of these legacy systems must be taken into account in developing architectures, data models, and interface specifications. On the other hand, the architecture should, as much as possible, enable the development of solutions that take advantage of new technologies. A tentative categorization for the discussion of architectural requirements is divided into the following categories:

- Learning Content Repository
- Course Management
- Courseware Execution
- Simulation Modes
- Courseware Development
- Content Submission and Testing
- Security

3.2 Architectural Elements

The section decomposes the overall Integrated Learning Environment (ILE) and PC simulation-based learning into its major component modules. The modules have been given generic names. Interfaces between the modules are also identified generically. The modules of the architecture are:

- Learning Content Management System (LCMS)
- Learning Management Systems (LMS)
- Student Learning Platforms – PCs (SLP-PC)
- Student Learning Platforms – PDAs (SLP-PDA)
- Web Browsers (WB)
- Web Browser Plug-Ins (WBPI)
- Course Management Systems (CMS)
- Stand-alone Simulators (SAS)
- Distributed Simulation Component (DSC)
- Course Authoring Systems (CAS)
- Course Testing Systems (CTS)

Figure 3.1 illustrates the overall simulation-based learning system architecture within the ILE and the relationships between the modules. The figure illustrates the relationship of a student's PC and/or PDA in the simulation-based learning environment. The Web Browser is the preferred mode of interaction with the Learning Management System (LMS) and Learning Content Management System (LCMS). A simulation could conceivably run as a Web Browser Plug-In (WBPI), as a Stand-Alone Simulator (SAS), or as a Distributed Simulation Component (DSC) that is connected to other simulation components running locally or on remote computers. DSCs could conceivably run on any computer system, given that appropriate security issues are addressed. Figure 3.1 illustrates that there are two possible paths for delivering content to the student's PC either directly from the LMS or indirectly from the LCMS. Typically the student's computer will interact with the LMS, but may have content subsequently delivered directly to the LCMS. The figure also shows the possibility of a Course Management System (CMS) running on the student's local computer. Although this is not the preferred mode of interaction, it may be necessary to use some commercially developed training materials. Also courseware development may occur directly through interactions with the LCMS or remotely on a separate Course Authoring System (CAS). In the second case, materials would be uploaded or physically transferred to the LCMS.

A more detailed discussion of each module and the interfaces are contained in the subsections that follow. Within each module subsection, a *summary description* and a list of *major functions* are provided. The specific commercial products that are currently used to implement the architecture are identified in Appendix C.

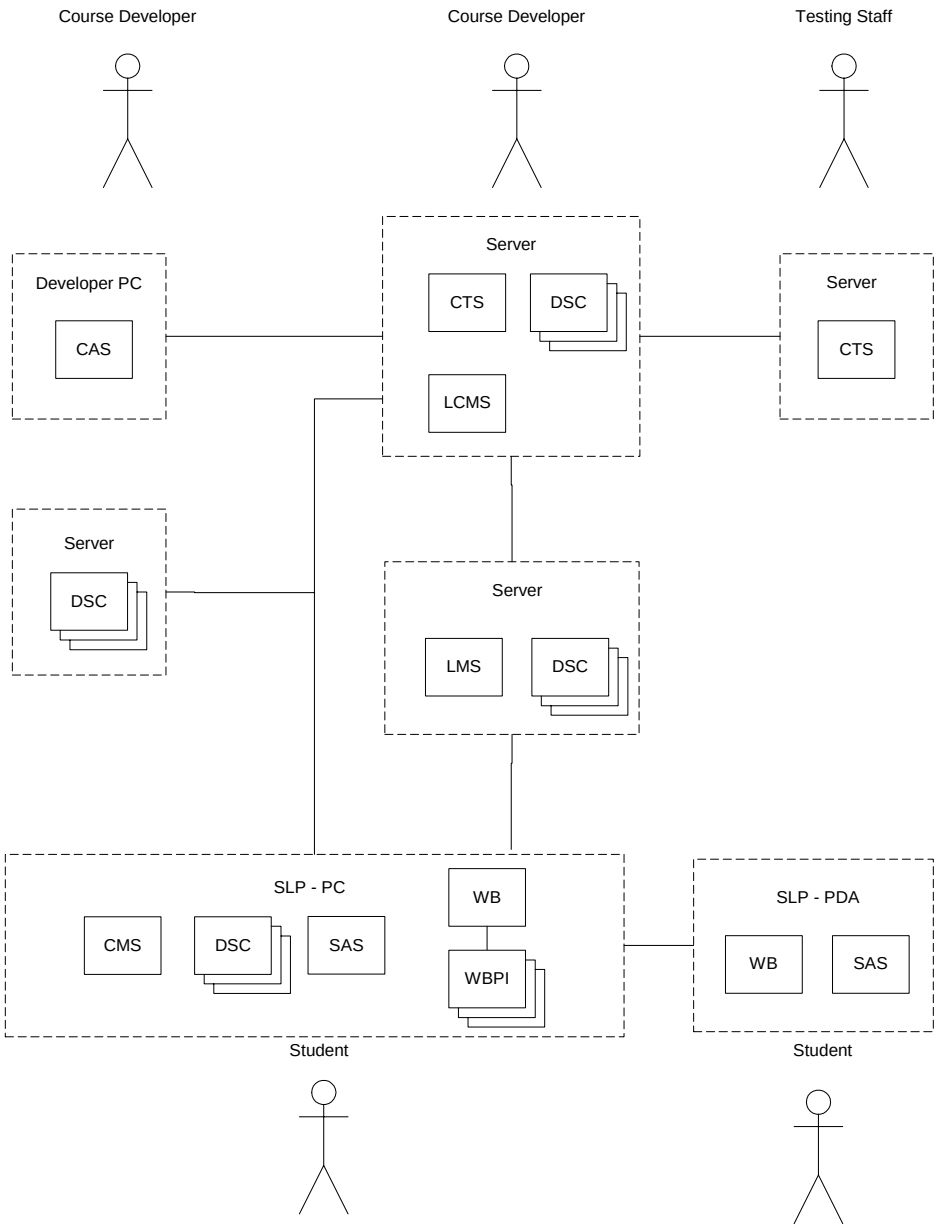

Figure 3.1 Simulation in the Integrated Learning Environment

3.2.1 Learning Content Management System (LCMS)

Summary Description: The LCMS is a development environment where multiple learning content authors can create, store, reuse, manage, and deliver digital learning content. The LCMS provides a central repository for the storage and retrieval of learning content objects. The LCMS may include Learning Management System (LMS) and Course Authoring System (CAS) functions. The LCMS may also interact with external LMS and CAS systems developed by other software vendors. See Appendix C for the current configuration of the Navy's LCMS.

Major Functions:

- Serve as a centralized repository for Reusable Learning Objects (RLO)
- Provide Learning Management System (LMS) functionality
- Provide Course Authoring System (CAS) functionality
- Provide network connectivity and interact with external Learning Management Systems (LMS) to deliver the course content to learners
- Provide network connectivity or media input options for external Course Authoring Systems (CAS) objects
- Provide a user interface for administering the repository including RLO tagging, cataloging and indexing, and searching
- Manage meta-data that describes the relevance of each learning object for specific learning objectives
- Provide authoring functions for creating learning objects in standard RLO formats
- Import and convert learning objects authored external to the system into RLO formats
- Provide authoring capabilities to create courses from the learning objects contained in the repository
- Manage workflow for RLO content creation
- Dynamically assemble personalized courses based to meet specific learning objectives

3.2.2 Learning Management System (LMS)

Summary Description: The LMS is a system for managing learners, keeping track of their progress and performance across all types of training activities. See Appendix C for the current Navy LMS products in use.

Major Functions:

- Publish a catalog of course offerings
- Provide communications connectivity and interact with the LCMS to obtain RLO course content
- Provide access control to courses including mechanisms for course enrollment and checking of prerequisites
- Manage personalize learning plans
- Launch and track learning applications
- Provide instructor interface for grading, retaking courses, setting up courses
- Maintain transcripts for the student population
- Take required tests and assessments online
- Track student progress, scores, completion, etc.
- Enable collaboration and communication between students and instructors
- Enable virtual classrooms
- Provide an API for interacting with courseware
- Launch and interact with executable course content
- Identify educational plans, measure and map skills, and perform gap analysis

3.2.3 Course Management System (CMS)

Summary Description: The preferred mechanism for course management is through the implementation of Web Browser (WB) and Web Browser Plug-Ins (WBPI). This module is provided as a placeholder for other CMS implementations. The primary purpose of the CMS is to provide testing, remediation, and tracking of the start and finish of various modules within a training course. A CMS is also known as Computer-Aided Instruction (CAI) or a Course Management Shell. The CMS typically provides collaboration tools such as asynchronous discussion forums, audio/video chat, text chat, email facilities. These capabilities are needed for courses that are conducted more effectively with instructor facilitation and student interaction. Most CMS products have LMS functions; however, they may or may not be compatible with the LMS. PC simulation vendors will be required to provide minimal CMS functions within their courseware to exchange information with the LMS module such as setting bookmarks, recording start and completion dates/times, grades, etc.

Major Functions:

- Run on the SLP-PC platforms to provide general courseware that does not use Web Browser functionality
- Provide communications connectivity and interact with the LMS to obtain RLO course content
- Record start and completion dates and times of course modules
- Provide collaboration tools such as chat and email facilities

3.2.4 Student Learning Platform – PC (SLP-PC)

Summary Description: The Student Learning Platform (SLP) module is the combination of PC hardware and software that allow students to access and execute PC simulation-based course content. Given the Navy's global mission and dispersed duty stations, systems will be deployed on a wide variety of hardware and software configurations at locations around the world. Student Learning Platforms (SLPs) include PC workstations, desktop and notebook computers. These computers may be government-owned or leased equipment. They may be a student's privately-owned computer, or those owned by Navy contractors and/or collaborators. As such, the configuration of SLPs **will not** always be under the control of the Navy. The Navy will establish configuration requirements for its SLPs. Non-Navy SLPs that meet or exceed Navy recommendations should be expected to function properly within this environment.

Appendix C provides a brief summary of configuration information for SLPs operating with the PC simulation-based learning environment. PC simulations provided by vendors would be so designed to be capable of deployment on the SLPs listed in the appendix. Differences in SLP deployment platforms can have an impact on fidelity, media selection, and interfaces with the LCMS, LMS, or CMS.

Major Functions:

- Serve as the platform for student execution of courseware for all supported learning modes (see Section 2.1.1)
- Satisfy minimum PC hardware and software compatibility requirements as described in ILE specifications (see Appendix C)
- Provide Web Browser (WB) compatibility for courseware execution
- Install Web Browser Plug-Ins (WBPI) as determined by specific courseware requirements through downloads or standard Navy distribution CD-ROMs
- Download and install Stand-alone Simulators (SAS)
- Provide network connectivity to the LMS and/or LCMS
- Download and store local copies of Re-usable Learning Objects (RLOs) used by courseware
- For non-networked PDAs, serve as interconnect point for PDA downloads and uploads

3.2.5 Student Learning Platform – PDA (SLP-PDA)

Summary Description: Serve as the learning platform on handheld Personal Digital Assistant (PDA) family of computers. The primary purpose of the PDA will be to enable informal on-the-job tutorials through the extension of on-line help capabilities. Simulation-based learning experiences occur as part of the process of performing a normal job assignment, such as system operation or maintenance. As a result of accessing an on-line "help" system via a laptop or PDA, a simulation-based course content object is activated to allow the individual to learn more about the operation or maintenance of the system.

Major Functions:

- Serve as the platform for student execution of selected courseware, as determined by PDA limitations
- Satisfy minimum PC hardware and software compatibility requirements as described in ILE specifications (see Appendix C)

- Provide Web Browser (WB) compatibility for courseware execution
- Install Web Browser Plug-Ins (WBPI) as determined by specific courseware requirements through downloads or standard Navy distribution CD-ROMs to linked PC platform
- Download and install Stand-alone Simulators (SAS)
- Provide indirect communication connectivity to the LMS and/or LCMS through PC link
- Download and store local copies of Re-usable Learning Objects (RLOs) used by courseware

3.2.6 Web Browsers (WB)

Summary Description: The Web Browser is the preferred mechanism for accessing course content over a network. The browser is responsible for interacting with the LMS or LMS capabilities within the LCMS to run the courseware. The Web Browser has no embedded simulation capability. Simulation capabilities will needed to be provided through plug-ins or through communications links to simulations running outside of the browser. The browser may be extended through plug-ins that provide multimedia, simulation, communications interconnection, or mobile code execution capabilities. See Web Browser Plug-Ins (WBPI), Stand-Alone Simulators (SAS), and Distributed Simulation Components (DSC) for more specifics on simulation execution. See Appendix C for approved browsers.

Major Functions:

- Provide Internet and Intranet access for courseware and simulations
- Access simulations running as Web Browser Plug-Ins (WBPI), Stand-Alone Simulators (SAS), or Distributed Simulation Components (DSC)
- Download and execute Reusable Learning Objects (RLOs)
- Run courseware and provide feedback to LMS/LCMS
- Install and manage plug-ins that extend browser capabilities

3.2.7 Web Browser Plug-ins (WBPI)

Summary Description: Web Browser Plug-Ins are software modules that extend the basic capabilities of the Web Browser. Multimedia players and mobile code interpreters are often implemented as plug-ins. Browser plug-ins can be downloaded and updated over the network. Security requirements may prohibit or require system administrator installation of plug-ins on Navy computers. Plug-ins could conceivably access and take advantage of any hardware, operating system, or software application capability on the client computer. As such, plug-ins could be used to construct virtually any kind of simulation, multimedia, communications, or mobile code execution functionality that is possible on a PC. See Appendix C for a current list of approved Web Browser Plug-ins.

Major Functions:

- Extend the basic capabilities of the Web Browser to include added courseware execution, multimedia, simulation, communications, or mobile code execution capabilities
- Implement RLO formats for multimedia interactions, simulation models, inputs, and outputs, wherever possible

3.2.8 Stand-alone Simulator (SAS)

Summary Description: The SAS is a placeholder in the architecture for simulations that cannot be implemented through web browser functionality. The SAS is a stand-alone engine for implementing simulations, i.e., models that represent activities and interactions over time. Simulation may be fully automated or it may be interactive or interruptible. The simulation is an operating representation of selected features of real-world or hypothetical events and processes. Stand-Alone Simulator refers to simulation executables that do not run or cannot run as a Web Browser Plug-In. This module includes simulations that simply run as a separate executable on a SLP-PC or SLP-PDA computer. These simulations may have been created commercially for use outside of the Navy. Cost or performance considerations may prevent migration of these simulations to the web-based simulation architecture. SAS data objects, e.g., inputs and models, should be stored in the LCMS wherever possible.

Major Functions:

- Provide specific types of simulation engine functionality
- Run a simulation that is an independent executable process on the PC platform with no direct communications link to the associated courseware
- Implement RLO formats for models, inputs, and outputs, wherever possible
- Provide user interfaces for setting up and interacting with the SAS
- Provide communications links to WB and WBPI courseware, wherever possible

3.2.9 Distributed Simulation Component (DSC)

Summary Description: The DSC is a module that enables the implementation and provides for access to and/or between distributed simulations. See Section 2.1.4.3 for background on distributed simulation. Three modes of distributed simulation are envisioned:

1) Client-server – the courseware running on the SLP-PC or the SLP-PDA is a client, it accesses a remote simulation running on the server
2) Modular decomposition – multiple simulation modules run on the student's local PC to form a single simulation for training purposes
3) multi-PC and/or multi-student simulation – simulation modules run across different platforms to implement multi-student exercises, provide performance, or functionality not possible on a single PC.

The DSC must address identification of remote modules, management of interactions between modules, communications interconnection, and in some cases, time synchronization, among other issues. The DSC may include other supporting modules that run as executables as part of a distributed simulation, for example HLA Run-time Infrastructure (HLA-RTI), CORBA middleware, or other communications services.

Major Functions:

- Provide specific types of simulation engine functionality
- Enable access to remote simulations running on servers
- Enable decomposition of large simulations into smaller modules
- Enable multi-platform execution of simulations

- Provide system clocks and time synchronization between simulation modules
- Provide communications interconnectivity between simulation modules
- Implement RLO formats for models, inputs, and outputs, wherever possible
- Provide user interfaces for setting up, visualizing, and interacting with the DSC
- Provide communications links to WB and WBPI courseware, wherever possible
- Provide control functions for managing the launching and managing the distributed execution of the simulation by instructors (and/or students, if appropriate)

3.2.10 Courseware Authoring System (CAS)

Summary Description: The CAS module is a placeholder in the architecture for any authoring systems other than the enterprise-wide LCMS selected by the Navy. It is not the preferred implementation scheme. The Navy's selected LCMS provides course-authoring functions that should maximize compatibility and interoperability with other modules in the architecture. Other authoring systems may be used to generate course content that is targeted for a broader customer base, i.e., use outside of the Navy community. If other systems are used to generate course content, their Re-usable Learning Objects (RLOs) will need to be exported and formatted for loading into the LCMS repository, wherever possible. Other CAS software may require proprietary Course Management Systems (CMS) to execute course content.

Major Functions:

- Create course content outside of the Navy's prescribed LCMS-based authoring environment
- Export RLOs in the Navy's prescribed formats
- Import RLOs from the Navy's LCMS to develop new or custom course content
- Deliver course content to non-Navy Course Management Systems (CMS)

3.2.11 Course Testing System (CTS)

Summary Description: This module includes all of the hardware, software, test suites, test cases, and test data used by the Navy for evaluating course content submissions. Tested submissions will include both custom courseware developed by Navy contractors and other general learning content obtained by the Navy for distribution through the ILE.

Major Functions:

- Ensure that all required RLOs have been submitted at the required level of granularity
- Test that RLOs conform to appropriate interface standards
- Test that RLOs download and perform properly in the web-based delivery environment on Student Learning Platform-PCs (SLP-PC), and if-appropriate, Student Learning Platform-PDAs
- Test that RLOs and mobile code conform to security requirements
- Test that RLOs only use authorized plug-ins or other support code
- Test that RLOs interact properly with the LCMS, LMS, and CMS with respect to administrative functions
- Test that web-based simulation-based RLOs perform according to specifications
- Test that Distributed Simulation Components (DSC) perform according to specifications
- Test that Stand-Alone Simulators (SAS) perform according to specifications

- Test that RLOs and simulators meet quality-of-experience requirements on specified hardware and software environments

3.2.12 System Interfaces

System interfaces will be required between each of the major modules that were identified in the preliminary system architecture. Each of these interfaces is briefly described below. Interfaces may be based on neutral standards, ad hoc standards, application programmer interface specifications, or custom interface specifications. The selected specifications or standards may evolve over time. The current selections can be found in Appendix D of this document.

- **RLO Data Formats** – This interface defines the levels of granularity and the data formats for all Re-usable Learning Objects (RLOs). RLOs should be:

  – Modular, free-standing, and transportable among applications and environments
  – Non-sequential, capable of being used in any order
  – Able to satisfy a single learning objective
  – Accessible to broad audiences (such that it can be adapted to audiences beyond the original target audience)
  – Coherent and unitary within a predetermined schema so that a limited number of meta-tags can capture the main idea or essence of the content
  – Unformatted so that they can be reused in different visual contexts without losing the essential value or meaning of the text, data, or images.

  The LCMS will enable the creation and retrieval of objects in the specified RLO format. The formats will be used by the LCMS to store objects in the repository. The RLOs will also be delivered to the LMSs and the Student Learning Platforms in this format. Other CAS applications will obtain objects in RLO format from the LCMS. Possible RLOs include course level, course objective, course topic, course lesson, course element (picture, animation, simulation, etc.).

- **LCMS to LMS API** – This application interface defines how the Learning Content Management System (LCMS) and Learning Management Systems (LMS) interact to locate course content and download RLOs.

- **WB and WBPI Courseware to LCMS/LMS Interface** – This interface defines how Web Browser and Web Browser Plug-In courseware applications interact with the LCMS/LMS to download RLO objects, mobile code, and SAS applications.

- **SLP-PC to SLP-PDA Interface** – This interface defines how Web Browser (WB) and Web Browser Plug-Ins (WBPI) running on a SLP-PDA interact with a SLP-PC to download RLOs, mobile code, and SAS applications.

  - **DSC Interface** – This interface defines how simulations will interact with courseware running in a Web Browser (WB) or a Web Browser Plug-In (WBPI). The interface defines how courseware will interact with a simulation running at a remote location, e.g., the LMS site. It also specifies how distributed simulations will interact with each other

and/or over the Internet or an Intranet. Distributed simulations may be used to implement classroom exercises where multiple students are interacting with the same executing simulation model. Multiple specifications and standards may be used to provide all required DSC interface functionality.

4. Simulation-based Training Using Game Technology

The Navy is considering the use of video game technology as the primary mechanism for implementing simulation-based training on PC family platforms. Game technology promises to provide a more engaging learning experience than traditional classroom or computer-aided instruction methods. Video game engines provide integrated environments for creating virtual worlds. They typically have capabilities for creating three-dimensional graphics, sound, animated characters, intelligent character behaviors, and various physical phenomena. They may also provide support for the creation of user level game modifications, Web-based software distribution, and distributed game participation over the Internet. This section describes how the student learning platforms and the simulators may be implemented using video game technology. Game engine technology may be used to implement the simulation modules introduced in the previous section of this document, i.e., Web Browser Plug-In (WBPI) simulators, Stand-alone Simulators (SAS), and/or Distributed Simulation Components (DSC).

Game technology has primarily been used for entertainment rather than educational purposes in the past. A number of changes will be required to support Navy needs. New functionality will need to be incorporated into game software to make it suitable for training applications. A common development strategy, a standard architecture, and neutral data interfaces will also be required to ensure that the Navy's "re-usability" objectives are met.

Software licenses for game development systems and game distribution are often quite expensive. Pervasive use of this technology will require that many Navy contractors will need access to licenses to develop learning applications. Perhaps hundreds of thousands of game-based training applications will ultimately be distributed. Game engine developers often collect royalties on each game sold. If commercial game engine software is used, the traditional business models of these software vendors may need to change.

This section briefly introduces topics relating to the application of video game technology to Navy training applications. Topics covered include a vision for game-based training system, example training applications, an overview of key system elements, and a recommended development approach. Background issues associated with the using this technology for Navy-learning applications is also briefly discussed.

4.1 Training System Vision

Video game technology could be used to create virtual environments for the Navy student. These environments would contain realistic three-dimensional graphics and sound that could significantly enhance the learning experience. Learning applications might address theory, operation, and maintenance of Navy systems. They also might include various operational scenarios where strategies and tactics are covered. Training could be developed for all branches of the Navy including aviation, surface, submariner, and shore-based activities. Officers and enlisted personnel alike could make effective use of these training capabilities. Professional growth needs for all career paths and ratings might eventually be addressed by this technology.

Elements of the video game-based training systems would include, where appropriate, real-time computer generated graphics and audio. Objects represented would include the environment,

ships, aircraft, vehicles, various systems, displays, tools, personnel, etc. Personnel would include various characters that represent the student, shipmates, friendly and opposing forces. Physics models and artificial intelligence would be used to give objects physically correct behaviors and movements, or enable them to act autonomously without human intervention.

The game environment would run on PC family computer systems including workstation, desktop, laptop, hand held, and PDA variety computers. The computers used to run the software could be Navy systems on ship and shore facilities or the student's personal computer at home. The software could also be used for classroom and team training at Navy schoolhouse facilities. The software engine, simulations, and other course content would all be delivered over the Internet via a Web browser interface. The software would install and run automatically without requiring special computer support expertise on the part of the student. Updates to software and course materials could be routinely disseminated over the Internet without resorting to the distribution of physical media, such as CD-ROMs.

The game engine, associated simulation modules, and Re-usable Learning Objects would be tested for security and certified by appropriate Navy testing facilities. The software would be secure and prevent the introduction of any security holes, viruses, worms, etc. onto the student's computer. The software also would not allow the student to achieve any unauthorized access to the host computer system areas or other networked systems as a result of the installation of the simulation-based training application.

Simulation-based learning applications could be developed for use in stand-alone mode or distributed multi-player mode to enable team training. Multi-student training applications would need distributed simulation capabilities to synchronize the software running on different platforms. Servers would be needed to store and distribute data to support these training exercises.

Simulations would be developed as part of training course materials and lesson plans. As such, the video game software will need to interface to the Navy's existing and evolving systems that handle instructional materials. The new learning applications will need to be adapted so that they can interface to existing Learning Content Management Systems (LCMSs) and Learning Management Systems (LMSs). These systems provide repositories for course materials, disseminate materials to students, and track progress, among other functions.

The Navy will need to keep costs down by re-using lessons, simulations, and graphical objects whenever possible. The strategy for achieving this objective based upon the concept of Re-Usable Learning Objects (RLOs). The simulation-based system will need standard data formats for these RLOs so that they can be used over and over again by different course content developers. The use of common game engines and game level editors will help ensure that RLOs are in fact reusable. To achieve Navy training objectives, course content developers will need to work to do most, if not all, of their creative work using game level editors, not performing low-level programming. Low-level programming, for example coding in C or C++, should be minimized to help ensure that applications developed by different course content providers are interoperable.

4.2 Example Training Applications

The objective of this section is to illustrate how game technology might be used to develop effective training applications. In game-based training, the student may have roles associated with specific Navy job functions. In stand-alone training, other roles would be assumed by non-player characters (NPC) programmed with intelligent, autonomous behaviors. If the objective were team training, other roles would be assumed by other students and be coordinated using distributed-simulation or multi-player game technology. Voice-over-Internet technology would be used to enable communication between players.

The game applications would be developed with realistic graphical and audio models of the environment, Navy vessels, aircraft, structures, characters, systems, tools, etc. Characters would be able to move around the virtual world performing actions as they would in the real associated Navy environment. Actions that the student's character could perform would be constrained by assigned roles and training objectives.

Some surface ship training applications will be briefly described to illustrate how this technology might be used. Similar examples could be developed for submarine, aviation, or shore-based training applications. On a surface ship, game-based training could be developed for:

- Bridge and deck operations
- Engineering plant operations
- Combat systems operations
- Weapon systems operations
- Air operations
- Damage control operations

Preventive and corrective maintenance of equipment in each operational area could also be addressed.

Taking the first area, bridge and deck operations, as an example. There are a number of scenarios that could be adapted to a game environment for both officer and enlisted personnel, e.g., conning officer, Officer-of-the-Deck, helmsman, etc. Possible evolutions include getting underway, piloting a ship in and out of port, performing navigation functions, demonstrating command and control in various underway evolutions, docking the ship, conducting man-overboard drills, carrying out underway replenishment, refueling, and highline details for personnel transfer. In the game environment, students would be responsible for setting up evolutions, using appropriate communications, issuing correct commands, making correct operational decisions, performing navigation, working with charts, flags, signaling, etc. This game-based training could be effectively used to teach nautical rules of the road, use of equipment (surface search radar for contacts and navigation), helmsmanship, etc.

4.3 Key Game System Elements

This section identifies some of the key elements that will be required if game technology is to be used for simulation-based training. An architecture and interface specification between

component modules is the subject of a separate document. Appendix E provides a brief overview of the neutral game engine architecture that is under development.

In the context of this document, the key game technology system elements are:

1) Graphics and sound
2) Game engine management
3) Character animation
4) Physics
5) Artificial intelligence
6) Game editor and development tools
7) Application data import and export
8) Multi-player operations and server support
9) Software distribution and security mechanisms
10) Learning system support

Each of these system elements is described in more detail in the sections that follow.

4.3.1 Graphics and Sound

Graphics and sound are the two primary outputs from game systems that are essential to create a realistic environment for the user. Graphics includes the capability to generate images from two and three-dimensional objects/scenes, map textures onto objects, provide various forms of lighting, cast shadows, model fog and smoke, provide cameras (viewing controls), among other features. Realism is further enhanced by the photographic quality, level of detail, the seamless joining of texture maps on surfaces, forms of randomness in images (dirt, rust, etc.), and other factors that typically distinguish a photograph from a computer-generated image. Graphics object libraries and procedurally-generated graphical objects (e.g., leaves on trees, large numbers of buildings in a city) can further enhance realism. The game engine typically provides interfaces to the computer's graphics card using OpenGL or Microsoft DirectX graphics software libraries.

Sound capabilities include the ability to play back multiple audio files simultaneously, create the effect of sounds emanating from specific locations, create ambient (background) noise, create special sound effects, and generation of realistic voices from text scripts.

4.3.2 Game Engine Management

Game engine management refers to the core of the system. It implements the mechanisms that support one or more game genres, e.g., role-playing, strategy, first person shooter, etc. It is responsible for execution control and supervisory capabilities that are built into the software. It can be thought of as the control logic or the program loop that runs the game. It is responsible for user interface management (input devices), displaying of multiple windows, etc.

The game engine is responsible for coordinating the other modules in the system. It is in part what distinguishes a game engine from a library of subroutines. It typically provides interfaces to a game engine editor, or level editor, that allows users/developers to create new game experiences at a higher level, avoiding low-level programming. It also may provide scripting

capabilities, i.e., the ability to write high-level programs that are interpreted by the game engine (as opposed to compiled code that is typically written in C or C++).

The game engine management module is responsible for scheduling or sequencing module execution to ensure that scene graphs get updated and graphics objects are rendered smoothly. It also ensures that updates to data, performance of physics calculations, etc. happen in a timely manner. As such, it is responsible for the system clock and calendar, the management of time, management of triggers, and the scheduling of events. It provides functions for saving and loading game levels and game state, pausing execution, playing back and rolling back game execution. It may manage the messaging scheme and databases by which communication is initiated between internal modules, and possibly updates to and from external modules (other game engines).

4.3.3 Character Animation

Functions of character animation include the development of character models (possibly using other character modeling software tools), definition of bones (controls for body segments), specification of range of motion for body joints, key frame animation, and the implementation of intelligent behavior. Intelligent behavior refers to the ability to perform complex sequence of activities that have been built up from basic motions using such techniques as finite state machines. Other measures of the quality of character animation include smooth interpolation of body joint regions when characters are in motion, blending of different motion sequences, footstep locomotion, motion capture capabilities (using actors outfitted with special hardware), modeling of the movement of cloth and hair, physics of character interaction (bumping into each other or falling down), manipulation of various props and objects. If motion capture is supported and any of several motion capture file formats may be used, e.g., BIP, CSM, and BVH files.

4.3.4 Physics

The physics module in a game system traditionally performs the calculations to enable realistic display of collisions, trajectories of projectiles, falling objects, cornering of vehicles, floating and sinking objects (e.g., vessels), movements of cloth and hair, etc. In the interest of keeping the number of categories reasonable, we include other physical, chemical processes and phenomena, such as environmental modeling (weather, sea conditions such as waves and swells), fires, the flow of liquids and gases, burn rates, behavior of electromagnetic and sound waves, etc. within the scope of the physics module. Proximity sensors are required by the physics module to implement interactions between objects.

Basic principles that are relevant to the implementation of physics modules include mass, center of mass, volume, Newton's laws, inertia, units of measure, geometry and vectors. Other concepts include kinematics of objects, linkages of objects, their linear and angular velocity, forces and torques acting on the objects, acceleration, and momentum in two and three dimensions. The physics module requires functions for performing physics calculations as well as data attributes on the objects to support those calculations. Soft body or rigid body mechanics may be used to model the behavior of objects in collision. Soft body mechanics will provide more realistic behavior with greater computational costs.

4.3.5 Artificial Intelligence

Artificial intelligence (AI) covers a number of different types of processing capabilities that contribute to the overall realism in the environment. Examples of AI processing capabilities include speech recognition, natural language processing, genetic algorithms and neural networks, randomness and statistical behaviors, rule-based systems, decision trees, goal-directed behavior, problem solving systems, fuzzy logic, and blackboard systems. AI may manifest itself in terms of path planning, strategies of non-player characters, coordinated behaviors, flocking, smart terrain, automatic solution of lower level behaviors for player characters, character motion, learning, etc. A discussion of all of these terms and concepts are beyond the scope of this document.

4.3.6 Game Editor and Development Tools

This characteristic area is given the highest weight of all the assessment criteria. The game editor and development tools are the primary way that learning content will be developed. This functionality must allow these contractors to work at a high level to develop re-usable learning content. It must provide capabilities to create environments, player and non-player characters, and supporting objects. It must provide a seamless interface to the game engine itself. It must allow the users to create behaviors on these characters and objects, define game levels, establish lesson plans, scoring schemes, etc. It needs to provide a scripting language to program these objects at a high level. It needs to provide previewing and debugging capabilities for testing scripts. The code must be well documented and easy to use. It should provide on-line help and a rich set of examples to help users get started. It needs to provide import and export capabilities to other common software packages, such as 3D Studio Max, to exchange graphics objects.

4.3.7 Application Data Import and Export

It is important the game engine support the import and export of data in as many appropriate standard and proprietary data formats as possible. Without these capabilities, developers have to translate their data into the formats that the engine supports. This is usually a costly and time-consuming process.

Relevant data formats include those for graphics files, video, audio, game levels, game state, character models, motion capture, RLOs, XML objects, etc. Graphics file include those to describe three-dimensional objects, textures, and motion capture. Most engines provide some support for 3D Studio Max and Maya formats. Other useful graphics file formats include JPEG, TIFF, and BMP. Multi-media and sound files also need to be supported, such as MPEG, WAV, and AVI files. The Navy's strategy is to build learning content based upon Reusable Learning Objects (RLOs) that are Sharable Content Object Reference Model (SCORM)-compliant and implemented using the Extensible Markup Language (XML). Traditional game software will probably not contain RLO, SCORM, or XML interfaces, but developers targeting simulation-based learning may offer those capabilities now or in the near future. Synthetic Environment Data Representation and Interchange Specification, now just commonly called SEDRIS, is a standard for representing environmental objects using XML. The Simulation-Based Learning policy guidelines document provides a more in depth discussion of data formats and the standards organizations that are responsible for them.

4.3.8 Multi-Player Operations and Server Support

The Navy would like to use simulation-based training applications in a team-training environment. This may include students training together in a classroom or working together as ad hoc teams across the Internet. Implementation of this training mode will involve implementation of distributed simulation capabilities using mechanisms such as the DoD's High Level Architecture (HLA) Run-Time Infrastructure (RTI), Distributed Interactive Simulation (DIS), Simulator Networking (SIMNET), or Massively Multi-Player (MMP) games. Voice Over Internet Protocol (VoIP), a mechanism for carrying out voice communications between players and instructors, may also be important in distributed simulation and gaming.

Major issues associated with distributed multi-player games are how and when players receive information on fellow players actions. Time lags may occur between when a player initiates an action and when other players see the action. This latency causes problems in the execution of distributed games.

The HLA RTI technology does not require the use of servers for centralized management of game data but uses time synchronization mechanisms that may be unacceptable in a game environment. In the HLA RTI world, simulators publish and subscribe to data objects to communicate. Simulations may be time-regulating or time-constrained. For more information on HLA technology, see (Kuhl, 1999).

MMP functionality involves the use of servers and is widely used in the gaming world. Due to its success as a commercial mechanism for distributed simulation and gaming, it should receive serious consideration for Navy applications. There have been security vulnerabilities associated with MMP games that have allowed players to cheat therefore appropriate safeguard must enacted. For more information on MMP technology, see (Alexander, 2003).

4.3.9 Software Distribution and Security Mechanisms

Due to the large numbers of potential users of simulation-based learning applications, the Navy would like to distribute the video game-based learning application and courseware over the Internet as web page downloads. The use of CD-ROMs or physical media of any sort must be avoided if at all possible. Furthermore, long download times during course execution and large software footprints on user machines are undesirable. Security constraints associated with the Navy Marine Corps Intranet (NMCI) environment must also be addressed. Use of Javascript is one programming approach that may have significant security flaws and probably should be avoided.

One possible solution would be to create a game engine that could be pre-tested for security flaws by an appropriate Navy facility. The game engine would be downloaded over the Internet and installed on the client computer automatically. Courseware and simulation-based learning objects would be downloaded before a training session or as they were needed. The client code would interact with the LCMS and/or LMS over the Internet using web-based protocols. Developers would create courses and associated content as Reusable Learning Objects to implement specific training needs. It is highly desirable that the game engine and the courseware have a small footprint on the client machine (i.e., use a minimal amount of disk space).

4.3.10 Learning System Support

Simulation-based learning applications that employ video game technology will need to be integrated with Learning Content Management Systems (LCMSs) and Learning Management Systems (LMSs). These interfaces will be needed to deliver course content to the student, manage instruction, and track the student's progress. Interfaces to LCMS and LMS software has not been a requirement for video game applications aimed at the entertainment market. As such, learning applications built on top of video game software will most likely need to be extended to support this interface requirement. Functions associated with learning system support will include: accessing content from an LCMS repository, organizing game content into lesson structures, supporting traditional instructional structures (courses, lessons, etc.), construction and varying of lesson scenarios for repeat execution, scoring progress against learning objectives, tracking student progress, providing on-line help associated with instructional programs, etc.

4.4 Development Approach

There is very little compatibility between different game engines on the market today. Game software is typically developed for a particular game engine, computer platform, or game console. Some of the game engines, level editors, and development systems allow game software to be ported to different computer operating systems or game consoles. Graphics editors, such as 3D Studio Max or Maya, are often used to create graphical objects that may be imported into almost any game engine, but that is where compatibility of data and code in the game world typically ends. The approach recommended in this document will help ensure greater software compatibility and re-usability of the training materials developed. It is based on the fact that many different game engine and course-content software developers may ultimately be involved in the process. Training materials will need to be developed and re-used over a time span of many years. Key aspects of the approach are the specification of an open architecture and neutral interfaces. The next two subsections address the issues of open source code and the major steps of the development process for simulation-based learning applications.

4.4.1 Open Source Code

Although most course material developers should be able to create content without resorting to low-level programming, game engine developers will need to create low-level code. If content developers need to write low-level code to develop course materials, there will undoubtedly be more compatibility, reliability, and security problems.

Access to source code will help ensure that the Navy will be able to make necessary changes to game engine software as technology evolves. Open source code should be created and used for the game engine, wherever possible. Some of the advantages of open source code are that other developers can see the code and make enhancements, there is less duplication of effort through the re-creation of the same code over and over, and increased collaboration opportunities to work on shared code, rather than proprietary products.

A NASA Technical Report provides excellent background and guidance on the issue of open source code; for more information see (NASA, 2003). Some of the key points from the report are excerpted below:

> *Open source refers to idea that the source code to an application is provided along with the executable code. … With open source code, there is the potential for improved*

software quality, more efficient software development, and increased collaboration. An organization known as the Open Source Initiative (OSI) provides the most widely recognized guidelines as to what constitutes Open Source; in particular the OSI provides guidance with respect to how to balance the intellectual property rights concerns of developers with openness (OSI, 2004). … The appeal of Open Source software for users is based on more than simply the desire to avoid paying money or to make a political statement. …Software that is available Open Source is easier to evaluate before making a deeper commitment. … In cases of closed or proprietary software, there may be no recourse for the user. Open Source software, on the other hand, leaves open the option of stepping in to keep the software alive. Enhanced collaboration would tend to produce software with more users, and in particular users who have a vested interest in seeing the software continue to thrive. In the long term those users may see value added in a commercialization that provides systems integration and support (think, for example, Linux). Enhanced dissemination would also tend to create a larger user base, and a large user base would enable more commercial opportunities. In the long term it may be worthwhile for the government to address software distribution, and in particular Open Source, in the initial request for proposals and statement of work.

In 2000, the President's Information Technology Advisory Committee (PITAC) was convened to address the subject of "Developing Open Source Software to Advance High End Computing," one of the committees recommendations was:

"*The Federal Government should allow open source development efforts to compete on a 'level playing field' with proprietary solutions in government procurement of high end computing software. Requests for Proposals (RFPs) from Federal agencies for high end computing software, tools, and libraries should include provisions allowing these efforts to be carried out using open source (PITAC, 2000).*"

A major issue with open source is the issue of software licensing. The topic is addressed in detail in the NASA report:

The GNU and Mozilla licenses require that derivative works be Open Source. This is potentially a valuable requirement for NASA, since it provides some assurance that the agency will have access to enhancements. This requirement also prevents certain types of "forking", where a user branches off from the NASA development in a closed way. In particular, this requirement would rule out cases where the branch goes proprietary. Weaker licenses, such as the BSD license, do not have this protection.

Example license agreements are contained in appendices C and E of the NASA report. In the case of NASA:

Software that is a joint work between NASA employees and NASA contractors is protected under copyright and, absent an agreement to the contrary, is co-owned by the U.S. Government and the contractor, with each having an independent right to use or license the use of the work with an obligation to account for royalties.

The groundwork established by GNU, Mozilla, and NASA in its report, should provide a basis for solving licensing issues associated with Navy game-based training applications. Licensing issues will require further investigation by government and industry lawyers.

4.4.2 Development Process Steps

Use of multiple developers should be encouraged to avoid the "vertical integration" of entire training applications using proprietary techniques. If the scope of responsibility for individual contractors, i.e., learning content providers, is carefully managed, the Navy can avoid becoming too dependent on a single vendor or that vendor's proprietary solutions. The steps of the proposed approach are outlined below:

- Identify best-available solutions from existing products and literature
- Develop a model that identifies Navy PC simulation-based training needs
- Specify system requirements, architecture, modules, and interfaces
- Establish learning object taxonomy, specifications, and repository
- Establish testing requirements, test system, and procedures
- Develop or acquire game engine modules from selected sources
- Develop simulation-based learning applications using game technology
- Pilot and deploy game-based training applications

Identify best-available solutions from existing products and literature – Commercial game engines, game software, published articles are identified and reviewed to determine the current state-of-the-art in gaming technology. Best-in-class solutions and technology directions are assessed. This assessment will help the Navy define requirements, determine what capabilities are available today and those that may be expected in the near future. It will also provide useful information for developing the game engine architecture, interface specifications, and detailed designs of component modules.

Develop a model that identifies Navy PC simulation-based training needs – To minimize the possibility of a piecemeal approach to simulation-based training application development and acquisition, a model that defines training needs will be developed. The purpose of the model is to create an integrated view of the game-based training applications that may eventually be needed. The model will identify Navy mission areas, objectives, systems, personnel, training needs, and associated game-based training opportunities. "Systems" actually refers to the complex network of systems that the Navy employs to carry out its mission, i.e., a systems-of-systems. Appropriate game genres will also be associated with training opportunities, e.g., role playing, strategy, etc. Unified Modeling Language (UML) use cases and system diagrams will be used to specify the model. The model will enable a building block approach to the development of training applications and re-usable learning objects. It will also be used to guide the development of the re-usable learning object taxonomy that will be discussed later in this section. The model will evolve over time, as training needs change.

Specify game engine system requirements, architecture, modules, and interfaces – Conduct requirements analyses and develop associated documents for Navy PC simulation-based learning applications that employ video game technology. Specify a system architecture for the game engine that decomposes it into major component modules. The architecture should be designed

to support the assembly of a game engine from component modules provided by different developers. Develop interface specifications for each module within the architecture. Interfaces will include the subroutine calls used by the modules to interact with each other and the data object structures used to pass information between the modules. Select existing standards where appropriate for the development of interfaces.

Establish learning object taxonomy, specifications, and repository – Reusable Learning Objects (RLOs) are a key element of the Navy's strategy to minimize the need for duplicative content development efforts. Objects contained within the game engine simulation environment should be re-usable learning objects. A classification scheme, or taxonomy, will be developed to classify these objects and define their attributes. The taxonomy and attribute structures will be developed in collaboration with subject matter experts from the various Navy user communities. Examples of objects in the taxonomy will include the environment (seas, ports, atmosphere, weather, navigation aids), ships, aircraft, vehicles, systems (navigation, weapons, electronics, engineering), tools, documents, messages, etc. A repository will be established for storing objects within the selected Learning Content Management System (LCMS). The taxonomy will be used to establish the data dictionary for the repository. Existing objects will be collected and translated for deposit in the repository. New objects will be created and archived, as needed, to support new instructional objectives.

Establish testing requirements, systems, procedures, and data sets – Success of the modular game engine architecture will be dependent upon a testing system that ensures that component modules comply with specifications. Testing requirements will be established that identify what functions each module must perform. A skeletal system will be established that can be used to evaluate new component modules that are being delivered to the Navy. The skeletal system will be used for testing. It will implement the module interface specifications and provide other functionality needed for testing that is not contained in the module under test. Procedures and data sets will be defined to ensure that tests are reliable and repeatable.

Develop or acquire game engine modules from selected sources – Game engine modules may either be developed from scratch or obtained from existing game engine developers. Access to source code will be a critical requirement for the long term. The Navy will need to ensure that it has access to make changes for security purposes and respond to the evolution technology, associated hardware and software. Open source solutions will be used wherever possible. Undoubtedly, an innovative approach will be needed to address licensing issues so that many potential Navy developers can get access to the engine.

Develop simulation-based learning applications using game technology – As simulation-based learning requirements are identified and resources become available, develop new training applications using game technology. The training applications will be traceable to the needs identified in the "needs model" discussed above. New applications will need to maximize use of Re-usable Learning Objects. Development activities may be partitioned among developers so as to improve efficiency and adherence to standard interfaces. If developers have the responsibility to build all elements of a training package, they may choose to create proprietary interfaces, e.g., an entire stand-alone executable computer-aided instruction package. If each developer's responsibilities are confined to particular modules or objects that are bounded by well-defined

interfaces, interoperability and re-usability will be more likely to be achieved. Efficiencies can also be obtained if one developer performs a limited set of functions over and over again, such as the creation of a library of objects to be used by other developers. For example, one developer might be given the responsibility of delivering a library of ship modules. Another might develop characters or models of shipboard electronics systems. An extension of this approach would be to use an assembly line as a model. Some developers would build basic components and implement their behaviors, such as electronics system models. Other developers would build sub-assemblies, such as combining components together to implement a ship's bridge, combat, or engineering areas. The next level of developer might assemble the objects into particular training courses, construct user interfaces, add multi-player features, define scenarios, etc.

Pilot and deploy game-based training applications – The final step of this development process is to transition game-based learning applications to the fleet. After the developer tests new courses, they would be transferred to a Navy testing facility for evaluation. If the software was found to function properly, meet security requirements, it could be transferred to a Navy schoolhouse facility for initial user testing using Intranet capabilities. If it was determined to work properly within those facilities, it may be deployed for Navy-wide use over the Internet.

5. Standards

With respect to PC simulation-based learning systems, the Navy is committed to moving towards industry standards and increasing the re-use of learning content objects. Future procurements will stress that proprietary solution and run-time engines must be avoided wherever possible – especially those requiring special licensing fees. In Information Technology Standards: Search for the Common Byte, Martin Libicki explains the role of standards in information systems:

> *"Standards solve particular problems, such as how to represent data efficiently or manage a communications system, and they create benefits—interoperability, portability, ease of use, expanded choice, and economies of scale—that only exist when many systems do things the same way."*

For more information on standards, see (Spivak and Brenner, 2001) and (Libicki, 1995). The remainder of this section identifies the organizations that are developing, or have developed, relevant standards and their associated specifications. This section does not address regulations concerning the development of software (e.g., Section 508 of the Rehabilitation Act regarding accessibility to computers for people with disabilities).

5.1 Standards Organizations

There are a number of organizations whose standards missions relate directly or indirectly to PC simulation-based learning. Some organizations have already resolved, or are currently working on, issues that may affect the Navy's objectives. The activities of some other organizations that are not currently involved in relevant standards should also be tracked, as they are likely to influence work in this area at a later time. The organizations are:

- Advanced Distributed Learning (ADL) Initiative
- American National Standards Institute (ANSI)
- Aviation Industry CBT (Computer-Based Training) Committee (AICC)
- Department of Defense (DoD)
- IMS Global Learning Consortium (IMS)
- Institute of Electrical and Electronics Engineers (IEEE)
- International Organization for Standardization (ISO)
- National Institute of Standards and Technology (NIST)
- Organization for the Advancement of Structured Information Standards (OASIS)
- Object Management Group (OMG)
- Synthetic Environment Data Representation and Interchange Specification (SEDRIS)
- Simulation Interoperability Standards Organization (SISO)
- Web 3D Consortium (WEB3D)
- World Wide Web Consortium (W3C)
- Other organizations

A brief introduction to each organization follows.

5.1.1 Advanced Distributed Learning (ADL) Initiative

The Advanced Distributed Learning (ADL) Initiative, sponsored by the Office of the Secretary of Defense (OSD), is a collaborative effort between government, industry and academia to establish a new distributed learning environment that permits the interoperability of learning tools and course content on a global scale. ADL's vision is to provide access to the highest quality education and training, tailored to individual needs, delivered cost-effectively anywhere and anytime. By working with industry and academia, the Department of Defense (DoD) is promoting collaboration in the development and adoption of tools, specifications, guidelines, policies, and prototypes that meet these functional requirements:

- Accessible from multiple remote locations through the use of meta-data and packaging standards
- Adaptable by tailoring instruction to the individual and organizational needs
- Affordable by increasing learning efficiency and productivity while reducing time and costs
- Durable across revisions of operating systems and software
- Interoperable across multiple tools and platforms
- Reusable through the design, management and distribution of tools and learning content across multiple applications.

The purpose of the ADL Initiative is to ensure access to high-quality education and training materials that can be tailored to individual learner needs and made available whenever and wherever they are required. The ADL Initiative is designed to accelerate large-scale development of dynamic and cost-effective learning software and systems to stimulate an efficient market for these products in order to meet the education and training needs of the Military Services and the nation's workforce of the future. ADL will achieve this through the development of a common technical framework for computer and net-based learning that will foster the creation of reusable learning content as "instructional objects." For further information on ADL, see http://www.adlnet.org/

5.1.2 American National Standards Institute (ANSI)

The American National Standards Institute (ANSI) is a private, non-profit organization (501(c)3) that administers and coordinates the U.S. voluntary standardization and conformity assessment system. The Institute's mission is to enhance both the global competitiveness of U.S. business and the U.S. quality of life by promoting and facilitating voluntary consensus standards and conformity assessment systems, and safeguarding their integrity. For more on ANSI, see http://www.ansi.org/.

5.1.3 Aviation Industry CBT (Computer-Based Training) Committee (AICC)

The Aviation Industry CBT (Computer-Based Training) Committee (AICC) is an international association of technology-based training professionals. The AICC develops guidelines for aviation industry in the development, delivery, and evaluation of CBT and related training technologies. The objectives of the AICC are as follows:

- Assist airplane operators in development of guidelines that promote the economic and effective implementation of computer-based training (CBT).

- Develop guidelines to enable interoperability.
- Provide an open forum for the discussion of CBT (and other) training technologies.

For further information on the AICC, see http://www.aicc.org/.

5.1.4 Department of Defense (DoD)

When there is an urgent need for a standard that has not been established by the national or international standards community, the Department of Defense or the services themselves may establish an internal standard. In some cases, the DoD standards may be adopted by the public sector, as was the case with the High Level Architecture. See Office Management and Budget Circular A-119 Revised 1998 on general government policy on creation of internal standards. With respect to simulation, the Department of Defense Defense Modeling and Simulation Office (DMSO) and the Navy Modeling and Simulation Offices have generated simulation standards for DOD use, e.g., the High Level Architecture (HLA) development was initiated by DMSO.

5.1.5 IMS Global Learning Consortium, Inc. (IMS)

IMS is a worldwide non-profit organization that includes more than 50 Contributing Members and affiliates. These members come from every sector of the global e-learning community. They include hardware and software vendors, educational institutions, publishers, government agencies, systems integrators, multimedia content providers, and other consortia. The Consortium provides a neutral forum in which members with competing business interests and different decision-making criteria collaborate to satisfy real-world requirements for interoperability and re-use.

The IMS Global Learning Consortium develops and promotes the adoption of open technical specifications for interoperable learning technology. IMS stands for Instructional Management Systems, a term that the organization has dropped due to the fact that it tended to raise more questions than it answered. Several IMS specifications have become worldwide de facto standards for delivering learning products and services. IMS specifications and related publications are made available to the public at no charge. No fee is required to implement the specifications. For more on IMS, see http://www.imsglobal.org/.

5.1.6 Institute of Electrical and Electronics Engineers, Inc. (IEEE)

The IEEE is a non-profit, technical professional association of more than 380,000 individual members in 150 countries. Through its members, the IEEE is a leading authority in technical areas ranging from computer engineering, biomedical technology and telecommunications, to electric power, aerospace and consumer electronics, among others. Through its technical publishing, conferences and consensus-based standards activities, the IEEE produces 30 percent of the world's published literature in electrical engineering, computers and control technology, holds annually more than 300 major conferences and has nearly 900 active standards with 700 under development.

The High Level Architecture (HLA) Working Group is an IEEE Standards Association (SA) Working Group and is under the government of the Simulation Interoperability Standards Committee (SISC), the IEEE Computer Society as well as the IEEE SA. The HLA has been developed with the objective of providing a common architecture applicable across all classes of

simulation to support simulation interoperability and reuse. The HLA is defined by the following three draft standards: Framework and Rules - IEEE Standard P1516, Federate Interface Specification - IEEE Standard P1516.1, Object Model Template (OMT) Specification - IEEE Standard P1516.2.

The Learning Technology Standards Committee (LTSC) is chartered by the IEEE Computer Society Standards Activity Board to develop accredited technical standards, recommended practices and guides for learning technology. The LTSC coordinates formally and informally with other organizations that produce specifications and standards for similar purposes. Working Groups within LTSC include:

- P1484.1 Architecture and Reference Model WG
- P1484.4 Digital Rights Expression Language (DREL) WG
- P1484.11 Computer Managed Instruction (CMI) WG
- P1484.12 Learning Objects Metadata (LOM) WG
- P1484.18 Platform and Media Profiles WG
- P1484.20 Competency Definitions WG

For further information on the IEEE, see http://www.ieee.org/.

5.1.7 International Organization for Standardization (ISO)

ISO (International Organization for Standardization) is the world's largest developer of standards. The ISO is a network of the national standards institutes of 147 countries, on the basis of one member per country, with a Central Secretariat in Geneva, Switzerland, that coordinates the system. ISO is a non-governmental organization: its members are not, as is the case in the United Nations system, delegations of national governments. Nevertheless, ISO occupies a special position between the public and private sectors. This is because, on the one hand, many of its member institutes are part of the governmental structure of their countries, or are mandated by their government. On the other hand, other members have their roots uniquely in the private sector, having been set up by national partnerships of industry associations.

Working through the ISO system, the sectors that need the standards initiate their development. When the need for a standard is felt by an industry or business sector, representatives of that sector communicate the requirement to one of ISO's national members. The latter then proposes the new work item to ISO as a whole. If accepted, the work item is assigned to an existing technical committee. Proposals may also be made to set up technical committees to cover new scopes of technological activity. In order to use resources most efficiently, ISO only launches the development of new standards for which there is clearly a market requirement.

For further information on the ISO, see http://www.iso.org/.

5.1.8 National Institute of Standards and Technology (NIST)

Founded in 1901, NIST is a non-regulatory federal agency within the U.S. Commerce Department's Technology Administration. NIST's mission is to develop and promote measurement, standards, and technology to enhance productivity, facilitate trade, and improve the quality of life. From automated teller machines and atomic clocks to mammograms and

semiconductors, innumerable products and services rely in some way on technology, measurement, and standards provided by the National Institute of Standards and Technology.

Within NIST, the Manufacturing Simulation and Visualization (MS&V) efforts are focused on accelerating the development of simulation standards. MS&V provides support to the Naval Education and Training Command in the area of PC simulation-based learning systems. For further information on NIST and MS&V, see http://www.nist.gov/ and http://www.mel.nist.gov/proj/msv.htm.

5.1.9 Organization for the Advancement of Structured Information Standards (OASIS)

OASIS is a not-for-profit, global consortium that drives the development, convergence and adoption of e-business standards. Members themselves set the OASIS technical agenda, using a lightweight, open process expressly designed to promote industry consensus and unite disparate efforts. OASIS produces worldwide standards for security, Web services, XML conformance, business transactions, electronic publishing, topic maps and interoperability within and between marketplaces.

OASIS has more than 600 corporate and individual members in 100 countries around the world. OASIS and the United Nations jointly sponsor ebXML (see http://www.ebxml.org/), a global framework for e-business data exchange. OASIS operates XML.org, a community clearinghouse for XML application schemas, vocabularies and related documents. For more on Oasis, see http://www.oasis-open.org/.

5.1.10 Object Management Group (OMG)

The Object Management Group (OMG) is an independent, not-for-profit corporation committed to developing technically excellent, commercially viable and vendor independent specifications for the software industry. The OMG was formed to create a component-based software marketplace by accelerating the introduction of standardized object software. Implementations of OMG specifications are used in many different industries worldwide. Specifications developed by the OMG include the Common Object Request Broker Architecture (CORBA), the Unified Modeling Language (UML), XML Metadata Interchange (XMI), and many other specifications for object services, Internet facilities, and domain specific interfaces. In the simulation domain the OMG has a specification called Facility for Distributed Simulation systems, which provides a CORBA interface for the services of the Run-time Infrastructure (RTI) of the High Level Architecture (HLA). For further information on the OMG and its standards, see http://www.omg.org/.

5.1.11 Synthetic Environment Data Representation and Interchange Specification (SEDRIS)

Environmental data is an integral part of many of today's information technology applications. The use of environmental data will grow substantially as availability and access to such data increases and as tools for manipulation of environmental data become less expensive and more sophisticated.

As this trend continues, the representation and sharing of environmental data will play a key role in the interoperation of heterogeneous systems and applications that use such data. This need was recognized in the mid-1980's, when the ability to network large numbers of heterogeneous simulation systems became a practical reality. Research and work in this area continued while a better and more complete understanding of the complex issues associated with describing and sharing of environmental data for a wide variety of (simulation) applications was formed. SEDRIS was conceived in order to tackle these issues in a uniform and unified manner.

Although the initial application domain for SEDRIS stems from the needs of the modeling and simulation field, it was immediately recognized that the representational technologies required to capture and communicate environmental data are fundamentally one and the same, and, in large part, can be dealt with independent of the end-applications.

At the same time, it was also understood that too often end-applications shape and form the characteristics of how data and data representation are used. The challenge for SEDRIS, then, was to provide a means for representation and sharing of environmental data that not only was efficient in practical use, but also was specific enough to address the real needs of a wide variety of end-applications, while preserving the degree of semantics needed for others to understand the nature of the data. The range of end-applications included representation of environmental data for such things as analysis, visualization, simulation, planning, modeling, etc. This took into account the meteorological and oceanographic communities, the simulation sector (both military and commercial), the GIS (or more broadly the environmental information systems) community, the military operational community, i.e., Command & Control, Communications, Computers and Intelligence (C4I), as well as others who needed to share or communicate environmental data.

Added to this was the goal of getting away from stovepipe views of the environment, and providing a mechanism that also allowed for integrated environmental data to be represented. Integrated environmental data, where ocean, terrain, atmosphere, and space data (about a region) can be seamlessly represented, was recognized as a key component of many future information technology applications. And although very few applications today deal with such diverse data at the same time, developers of SEDRIS believed such a need would be reality in the future.

With these objectives and challenges in mind, SEDRIS was initiated in 1994. SEDRIS has been conducted as an open project with the objective of solving these challenges in practical ways that can be immediately used by both data providers and consumers, while at the same time leveraging and taking advantage of existing standards whenever possible. For further information on SEDRIS, see http://www.sedris.org.

5.1.12 Simulation Interoperability Standards Organization (SISO)

SISO focuses on facilitating simulation interoperability and component reuse across the DoD, other government, and non-government applications and seeks to serve the broad Modeling and Simulation (M&S) community by:

- Providing a forum for the interchange of new ideas, concepts, and technology through the Simulation Interoperability Workshops and the Conference on Computer Generated Forces and Behavioral Representation.

- Educating M&S practitioners and sponsors regarding implementation through tutorials held at the Workshops and through the quarterly publication of an on-line technical magazine titled "Simulation Technology."
- Supporting the development of standards, practices, and guides for use in various applications.

The Simulation Interoperability Standards Organization (SISO) originated over ten years ago with a small conference held April 26 and 27, 1989, called, "Interactive Networked Simulation for Training." The original conference attracted approximately 60 people. The group was concerned that there was activity occurring in networked simulation, but that it was occurring in isolation. The group believed that if there were a means to exchange information between companies and groups that the technology would advance more rapidly. The group also believed that once the technology begins to stabilize then there would also be a need for standardization. The technology and the consensus of the community would be captured in the standards as networking or simulation technology matured.

SISO evolved from the Distributed Interactive Simulation (DIS) Workshops and is now focused on creating standards based on the major project SIMNET that was established as the baseline standard from which to move forward. In late 1996, in light of the development of the High Level Architecture (HLA), the DIS organization transformed itself into a more functional organization called SISO. SISO currently has 1176 official members representing over 400 organizations including 51% commercial, 41% government/military and 8% academia. The members represent 12 countries including Canada, France, Germany, Ghana, Israel, Japan, Netherlands, Scotland, Spain, Sweden, United Kingdom and the USA. For further information on SISO, see http://www.sisostds.org/.

5.1.13 Web 3D Consortium (Web 3D)

The Web3D Consortium, Inc. is a nonprofit corporation with the mission of fostering and evangelizing all three-dimensional technologies on the Internet. The Web3D Consortium was formed to provide a forum for the creation of open standards for Web3D specifications, and to accelerate the worldwide demand for products based on these standards through the sponsorship of market and user education programs. The Web3D Consortium's goals are to:

- Foster the ongoing development of three dimensional specifications and standards on the Internet.
- Promote rapid industry adoption of and adherence to the X3D specification.
- Offer opportunities for the three-dimensional development community to meet and cooperate in the evolution of Web3D technologies.
- Educate the business and consumer communities on the value, benefits and applications of three-dimensional technologies on the Internet.
- Support the creation of conformance tests to assure Web3D interoperability.
- Liaison with educational institutions, government research institutes, technology consortia, and other organizations that support and contribute to the development of specifications and standards for Web3D.

Today, the Web3D Consortium is utilizing its 1500-2000 strong Internet development community, and its broad-based industry support to systematically move the VRML97 ISO Standard forward. This community has spearheaded the development of the VRML 1.0 and 2.0 specifications, which provide the basis for the development of associated applications. The organizations involved in this effort felt that the creation of an open consortium focused exclusively on Web3D would provide the structure necessary to stabilize, standardize, and nurture the technology for the entire community.

Prominent technical activities include the Extensible 3D (X3D) specification, which is extending VRML97, using the Extensible Markup Language (XML). Through the well-coordinated efforts of dedicated Working and Task Groups and ISO/W3C Liaisons, the Web3D Consortium is maintaining and extending its standardization activities well into the next Millennium. For further information on Web3D, see http://www.web3d.org/.

5.1.14 World Wide Web Consortium (W3C)

The World Wide Web Consortium (W3C) develops interoperable technologies (specifications, guidelines, software, and tools) to lead the Web to its full potential. W3C is a forum for information, commerce, communication, and collective understanding. The World Wide Web Consortium was created in October 1994 to lead the World Wide Web to its full potential by developing common protocols that promote its evolution and ensure its interoperability. W3C has about 400 member organizations from all over the world and has earned international recognition for its contributions to the growth of the Web.

W3C's long-term goals for the Web are:

- *Universal Access:* To make the Web accessible to all by promoting technologies that take into account the vast differences in culture, languages, education, ability, material resources, access devices, and physical limitations of users on all continents.
- *Semantic Web:* To develop a software environment that permits each user to make the best use of the resources available on the Web.
- *Web of Trust:* To guide the Web's development with careful consideration for the novel legal, commercial, and social issues raised by this technology.

W3C contributes to efforts to standardize Web technologies by producing specifications (called "Recommendations") that describe the building blocks of the Web. W3C makes these Recommendations (and other technical reports) freely available to all.

Development of XML started in 1996 and it has been a W3C Recommendation since February 1998. Extensible Markup Language (XML) is a simple flexible text format derived from the Standard Generalized Markup Language (SGML). Originally designed to meet the challenges of large-scale electronic publishing, XML is also playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere (see Section 5.2.1). For further information on W3C, see http://www.w3.org/.

5.2 Standards Specifications

What relevant standards specifications have been created by these organizations? Current standards that most directly affect the implementation of PC simulation-based learning systems fall into the following categories:

- Extensible Markup Language (XML) standards
- Simulation standards
- Multimedia standards
- Product representation standards
- Knowledge representation standards
- Programming language standards
- Other standards

Within the standards categories, each standard and its significance is briefly discussed. See Appendix D for a list of standards currently selected for implementation of PC simulation-based learning systems.

5.2.1 Extensible Markup Language Standards

A number of standards are being developed using XML. Some of the specifications that are most relevant to Navy PC simulation-based learning applications follow.

5.2.1.1 Extensible Markup Language (XML)

The eXtensible Markup Language (XML) was developed by the W3C to allow the definition of simple structured text data formats. It is derived from the SGML, the ISO standard language used extensively in the publishing domain. The focus of XML though is not publishing. It facilitates the creating of structured documents that are can be read by humans or processed by machines. XML accomplishes this by providing mechanisms to add markup to "raw" data in standardized way. Markup, in the form of tags or other predefined character sequences, provides a means defining the layout and logical structuring for a document. Since XML allows for the definition of user-defined tags, not only structuring but also semantic information can be added as well. This is more flexible and differs from a related and widely used markup language, the Hypertext Markup Language (HTML), where all of the tags used to markup up a document are pre-defined.

There are may other standards developed by the W3C that are either derived from or work with XML. These include but are not limited to:

- XML Path Language (Xpath) - A method for referencing parts of an XML document)
- Document Object Model (DOM) – a method for creating or manipulating XML documents
- Extensible Stylesheet Language Transformations (XSLT) – a language for transforming an XML document into another XML document
- XML Schema – an XML based language that is used to define the format and semantics of an XML document.

There are many other standards related to XML supported by the W3C. For further information, see http://www.w3.org/.

5.2.1.2 Department of the Navy (DON) XML

The Department of the Navy recognizes the value of XML as a powerful enabler of enterprise IT interoperability. The Department of the Navy Chief Information Officer (DON CIO) has chartered the DONXML Work Group with achieving the Department's vision of fully exploiting XML as an enabling technology to achieve interoperability in support of maritime information superiority. The DONXML Work Group is currently in the process of developing the following deliverables:

- DON XML Vision document (completed)
- DON XML Developer's Guide (version 1.1 released)
- DON XML Policy, Procedures, and Governance Structure
- DON XML Implementation Plan
- DON XML Registry/Repository Requirements

For further information, see the DON XML website at https://quickplace.hq.navy.mil/navyxml/.

5.2.1.3 SEDRIS

SEDRIS is fundamentally about two key aspects: (1) representation of environmental data, and (2) the interchange of environmental data sets. To achieve these representation and interchange objectives, SEDRIS relies on its five core technology components. These are the SEDRIS Data Representation Model (DRM), the Environmental Data Coding Specification (EDCS), the Spatial Reference Model (SRM), the SEDRIS interface specification (API), and the SEDRIS Transmittal Format (STF).

Because of these characteristics, the representational aspect of SEDRIS is much like a language or a method for unambiguously describing the environment, independent of whether the environment is geo-specific, geo-typical, or completely fictitious. And the interchange aspect is a mechanism for sharing the described environmental data. Put together, SEDRIS is simply an infrastructure technology. It provides the enabling foundation for IT applications to express, understand, share, and reuse environmental data. For further information on SEDRIS, see http://www.sedris.org.

5.2.1.4 Shareable Content Object Reference Model (SCORM)

The SCORM defines a Web-based learning "Content Aggregation Model" and "Run-Time Environment" for learning objects. SCORM is a collection of specifications adapted from multiple sources to provide a comprehensive suite of e-learning capabilities that enable interoperability, accessibility and reusability of Web-based learning content. The work of the ADL Initiative to develop the SCORM is also a process to knit together disparate groups and interests. This reference model aims to coordinate emerging technologies with commercial and/or public implementations.

The SCORM applies current technology developments to a specific content model by producing

recommendations for consistent implementations by the vendor community. It is built upon the work of the AICC, IMS, IEEE, ARIADNE and others to create one unified "reference model" of interrelated technical specifications and guidelines designed to meet DoD's high-level requirements for Web-based learning content. The SCORM includes aspects that affect learning management systems and content authoring tool vendors, instructional designers and content developers, training providers and others.

The following SCORM specifications are available as downloads from the ADL web site.

- Sharable Content Object Reference Model (SCORM) Version 1.2
- SCORM Version 1.2 Addendums
- SCORM Version 1.3 Application Profile Working Draft Version 1.0
- SCORM Version 1.2 Conformance Requirements Version 1.2
- SCORM Version 1.2 XML Controlling Document - Advanced Distributed Learning SCORM Version 1.2 Content Packaging XML XSD
- SCORM Version 1.2 XML Controlling Document - IMS Global Learning Consortium, Inc. Learning Resource Meta-data Specification Version 1.2.1 XML XSD
- SCORM Version 1.2 XML Controlling Document - IMS Global Learning Consortium, Inc. Content Packaging Specification Version 1.1.2 XML XSD

5.2.2 Simulation Standards and Specifications

In the area of simulation, recent standards and specifications activities (HLA and XMSF) have focused on the creation of distributed simulations. Another specification, DEVS focuses on a neutral format for representing discrete event models.

5.2.2.1 High Level Architecture (HLA)

Framework and Rules - IEEE Standard P1516: The HLA rules describe the responsibilities of federates (simulations, supporting utilities, or interfaces to live systems) and federations (sets of federates working together to support distributed applications). The rules comprise a set of underlying technical principles for the HLA. For federations, the rules address the requirement for a federation object model (FOM), object ownership and representation, and data exchange. For federates, the rules require a simulation object model (SOM), time management in accordance with the HLA Runtime Infrastructure (RTI) time management services, and certain required functionality and constraints on attribute ownership and updates.

Federate Interface Specification - IEEE Standard P1516.1: In the HLA, federates interact with an RTI (analogous to a special-purpose-distributed operating system) to establish and maintain a federation and to support efficient information exchange among simulations and other federates. The HLA interface specification defines the nature of these interactions, which are arranged into sets of basic RTI services.

Object Model Template (OMT) Specification - IEEE Standard P1516.2: The HLA requires simulations (and other federates) and federations to each have an object model describing the entities, not necessarily platform entities, represented in the simulations and the data to be exchanged across the federation. The HLA object model template prescribes the method for recording the information in the object models, to include objects, attributes, interactions, and

parameters, but it does not define the specific data (e.g., vehicles, unit types) that will appear in the object models.

5.2.2.2 Extensible Modeling and Simulation Framework (XMSF)

The EXtensible Modeling and Simulation Framework (XMSF) provides a framework which allows both Department of Defense (DoD) and non-DoD Modeling and Simulation (M&S) projects to take advantage of Web-based technologies. Such a framework aids M&S applications to interoperate, as well as enable M&S development. XMSF is a community initiative to enable interoperability and composability. The XMSF partners are working with multiple organizations in the M&S community of interest including SISO, Web3D, OMG, and many commercial and DoD stakeholders. The initial XMSF exemplar uses web-based communication protocols, SOAP and BEEP, to allow a High Level Architecture (HLA) compliant simulation to communicate with the DMSO/SAIC Run Time Infrastructure (RTI) over the Web.

5.2.2.3 Discrete Event System Specification (DEVS)

The Discrete Event System Specification (DEVS) is for specifying a mathematical object called a system. Elements of a DEVS system include a time base, inputs, states, outputs, and functions for determining next states, given current states and inputs. The initial work on DEVS was funded by the Defense Advanced Research Projects Agency (DARPA). A standards product development group is planned within the SISO organization to promote DEVS to standards status, for more information, see www.sisostds.org or http://www.acims.arizona.edu/.

5.2.3 Multimedia Standards

Various graphics and audio formats will be critical the implementation and viewing of PC simulations. A few key standards are described below. In addition to the standard multimedia image and sound file formats prescribed by the Navy for other learning applications, the following multimedia formats will be relevant to simulation, see NMCI Gold Disk at http://www.nmci-isf.com/downloads/Gold_disk_contents.pdf for a list of approved multimedia formats. For more on graphics file formats, see (Murray and VanRyper, 1996).

5.2.3.1 Virtual Reality Modeling Language (VRML)/X3D

The Virtual Reality Modeling Language (VRML) is a file format for describing interactive 3D objects and worlds. VRML can be used on the Internet, Intranets, and local client systems. VRML is also intended to be a universal interchange format for integrated 3D graphics and multimedia. VRML may be used in a variety of application areas such as engineering and scientific visualization, multimedia presentations, entertainment and educational titles, web pages, and shared virtual worlds. VRML is capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. VRML browsers, as well as authoring tools for the creation of VRML files, are widely available for many different platforms. VRML supports an extensibility model that allows new dynamic 3D objects to be defined allowing application communities to develop interoperable extensions to the base standard. There are mappings between VRML objects and commonly used 3D application programmer interface (API) features. To view a VRML file, you need a VRML viewer or browser, which can be a plug-in for a Web browser. For more on VRML, see http://www.web3d.org/fs_technicalinfo.htm.

5.2.3.2 H-Anim

As the 3D Internet continues to grow, there will be an increasing need to represent human beings in online virtual environments. Achieving that goal will require the creation of libraries of interchangeable humanoids, as well as authoring tools that make it easy to create new humanoids and animate them in various ways. H-Anim specifies a standard way of representing humanoids in VRML 2.0. This standard will allow humanoids created using authoring tools from one vendor to be animated using tools from another. VRML humanoids can be animated using key-framing, inverse kinematics, performance animation systems and other techniques.

H-Anim design goals are as follows:

- **Compatibility:** Humanoids should work in any VRML 2.0 compliant browser.
- **Flexibility:** No assumptions are made about the types of applications that will use humanoids.
- **Simplicity:** When in doubt, leave it out. The specification can always be extended later.

Much of the design is driven by these three goals. The compatibility requirement has led to avoiding the use of scripting, since the VRML 2.0 specification does not require any particular scripting language to be implemented. The flexibility requirement has led to making the specification fairly "low-level", in that it allows direct access to the joints of the humanoid's body. The simplicity requirement has led to focusing specifically on humanoids, instead of trying to deal with arbitrary articulated figures.

The human body consists of a number of *segments* (such as the forearm, hand and foot) that are connected to each other by *joints* (such as the elbow, wrist, and ankle). In order for an application to animate a humanoid, it needs to obtain access to the joints and alter the joint angles. The application may also need to retrieve information about such things as joint limits and segment masses.

A VRML Humanoid file contains a set of Joint nodes that are arranged to form a hierarchy. Each Joint node can contain other Joint nodes, and may also contain a Segment node that describes the body part associated with that joint. The file also contains a single Humanoid node that stores human-readable data about the humanoid such as author and copyright information. That node also stores references to all the Joint and Segment nodes. Additional nodes can optionally be included in the file, such as Viewpoints and possibly a NavigationInfo node.

Key-frame animation sequences can be stored in the same file, with the outputs of various VRML interpolator nodes being ROUTEd to the joints of the body. Alternatively, the file may include Script nodes that access the joints directly. In addition, applications can obtain references to the individual joints and segments from the Humanoid node. Such applications will typically animate the humanoid by setting the joint rotations. For further information, see http://www.web3d.org/. H-Anim is also moving forward as a separate ISO standard, ISO/IEC FCD 19774 — Humanoid animation.

5.2.4 Product Representation Standards

It is likely that a number of future PC simulation-based training courses will involve learning the theory, operation, and maintenance of Navy equipment. The following standards are vendor-independent formats for the delivery of part drawings and product specification data. These data formats may be used to obtain information that is used to create RLOs in the future.

5.2.4.1 Standard for the Exchange of Product Model Data (STEP)

The information generated about a product during its design, manufacture, use, maintenance, and disposal is used for many purposes during that life cycle. The use may involve many computer systems, including some that may be located in different organizations. In order to support such uses, organizations need to be able to represent their product information in a common computer-interpretable form that is required to remain complete and consistent when exchanged among different computer systems.

ISO 10303, informally known as STEP, is an International Standard for the computer-interpretable representation and exchange of product data. The objective is to provide a mechanism that is capable of describing product data throughout the life cycle of a product, independent from any particular system. The nature of this description makes it suitable not only for neutral file exchange, but also as a basis for implementing and sharing product databases and archiving. The structure of this International Standard is described in ISO 10303-1. The numbering of the parts of this International Standard reflects its structure:

- Description methods (Parts 1 - 19)
- Implementation methods (Parts 20 - 29)
- Conformance testing methodology and framework (Parts 30 - 39)
- Integrated generic resources (Parts 40 - 49)
- Integrated application resources (Parts 100 - 199)
- Application protocols (Parts 200 - 299)
- Abstract test suites (Parts 300 - 399)
- Application interpreted constructs (Parts 500 - 599)
- Application Modules (Parts 1000 -)

For more information on STEP, see http://www.iso.org/. or http://www.tc184-sc4.org/SC4_Open/SC4_Work_Products_Documents/STEP_(10303)/.

5.2.4.2 Initial Graphics Exchange Specification (IGES)

IGES defines a neutral data format that allows for the digital exchange of information among computer-aided design (CAD) systems. CAD systems are in use today in increasing numbers for applications in all phases of the design, analysis, manufacture and testing of products. Since it is common practice for a designer to use one supplier's CAD system and for the contractor and subcontractors use different systems, there is a need for the ability to exchange data digitally among all CAD systems.

IGES provides a neutral definition and format for the exchange of specific data. Using IGES, a user can exchange product data models in the form of wire frame or solid representations as well

as surface representations. Applications supported by IGES include traditional engineering drawings as well as models for analysis and/or various manufacturing functions. In addition to the general specification, IGES includes application protocols in which the standard is interpreted to meet discipline specific requirements.

IGES is an American National Standard (ANS). Version 1.0 of the specification was adopted as an American National Standard ANS Y14.26M-1981in November of 1981. Subsequent versions have followed as ANS specifications with version 5.2 approved and adopted by the American National Standards Institute (ANSI) in 1993. The current version, IGES 5.3, was approved by ANSI under the guidelines of the U.S. Product Data Association (US PRO) during September 1996. The latest version of the IGES standard is designated ANS US PRO/IPO-100-1996. For more on IGES, see https://uspro.aticorp.org/.

5.2.5 Knowledge Representation Standards

As PC simulation-based learning applications evolve, representation of knowledge will become more important two standards that are relevant to this area are briefly described below.

5.2.5.1 Knowledge Interchange Format (KIF)

Knowledge Interchange Format (KIF) is a computer-oriented language for the interchange of knowledge among disparate programs. It has declarative semantics (i.e., the meaning of expressions in the representation can be understood without appeal to an interpreter for manipulating those expressions). It is logically comprehensive (i.e., it provides for the expression of arbitrary sentences in the first-order predicate calculus); it provides for the representation of knowledge about the representation of knowledge; it provides for the representation of non-monotonic reasoning rules; and it provides for the definition of objects, functions, and relations. Standardization activities for KIF are now part of the ISO JTC 1 SC 32 Data Management and Interchange efforts called “Common Logic (CL) - A Framework for Family of Logic-Based Languages,” see www.jtc1sc32.org or http://logic.stanford.edu/kif/kif.html.

5.2.5.2 Web Ontology Language (OWL)

The Web Ontology Language (OWL) is a semantic markup language for publishing and sharing ontologies on the World Wide Web. OWL is developed as a vocabulary extension of RDF (the Resource Description Framework) and is derived from the DAML+OIL Web Ontology Language. The OWL specification document contains a structured informal description of the full set of OWL language constructs and is meant to serve as a reference for OWL users who want to construct OWL ontologies. For more on OWL, see http://www.w3.org/TR/owl-ref/

5.2.6 Mobile Code Programming Language Standards

This section identifies some of the mobile code programming languages. Due to the security problems these languages may create, their use for Navy training applications may be restricted. See appropriate guidelines for information on which languages are currently approved for use. The NMCI Solicitation, Attachment 4, Paragraphs 1.2.3 and 1.2.4, specifies that both ActiveX and JavaScript will be completely blocked, see https://nmci.spawar.navy.mil/cl_solicitation_docs.html**.**

5.2.6.1 Java

Java is a programming language expressly designed for use in the distributed environment of the Internet. It was designed to have the "look and feel" of the C++ language, but it is simpler to use than C++ and enforces an object-oriented programming model. Java can be used to create complete applications that may run on a single computer or be distributed among servers and clients in a network. It can also be used to build a small application module or applet for use as part of a Web page. Applets make it possible for a Web page user to interact with the page. For more information see http://www.sun.com/java.

5.2.6.2 JavaScript/Jscript/ECMAScript

A scripting language developed by Netscape to enable Web authors to design interactive sites. Although it shares many of the features and structures of the full Java language, it was developed independently. JavaScript can interact with HTML source code, enabling Web authors to spice up their sites with dynamic content. JavaScript is endorsed by a number of software companies and is an open language that anyone can use without purchasing a license. Recent browsers from Netscape and Microsoft support Javascript.

ECMAscript (ECMA standard 262, ISO standard 16262) is the standardized version of the core JavaScript language. Microsoft's implementation of ECMAscript is called JScript. JScript only runs in Microsoft's Internet Explorer browser.

5.2.6.3 VBScript

VbScript is Microsoft's scripting language that is an extension of their Visual Basic language. VBScript can be used with Microsoft Office applications and others. It can also be embedded in web pages but can only be understood by Internet Explorer. The Visual Basic language is a BASIC variant with object-oriented features. Objects may include applications, windows and selections.

5.2.6.4 Perl

Short for Practical Extraction and Report Language, Perl is a programming language especially designed for processing text. Because of its strong text processing abilities, Perl has become one of the most popular languages for writing CGI scripts. Perl is an interpretive language, which makes it easy to build and test simple programs.

5.2.6.5 PHP: Hypertext Preprocessor (PHP)

Self-referentially short for PHP: Hypertext Preprocessor, an open source, server-side, HTML embedded scripting language used to create dynamic Web pages. In an HTML document, PHP script, which has a similar syntax to that of Perl or C, is enclosed within special PHP tags. Because PHP is embedded within tags, the author can jump between HTML and PHP instead of having to rely on heavy amounts of code to output HTML. And, because PHP is executed on the server, the client cannot view the PHP code. PHP can perform any task that any CGI program can do, but its strength lies in its compatibility with many types of databases.

5.2.7 Other Standards and Specifications

5.2.7.1 AICC Guidelines and Recommendations (AGRs)

The AICC has published a number of guidelines and recommendations that could be used to implement PC simulation-based learning applications:

- AGR 001 AICC Publications - Summarizes all of the publications issued by the AICC. It identifies and provides an abstract of current AICC Guidelines and Recommendations, technical documents, and white papers.

- AGR 002 Courseware Delivery Stations - Contains recommendations to the aviation industry for the acquisition of a computer-based training student delivery station. Recommendations, with accompanying rationale, are provided for a delivery station's CPU, clock speed, bus, power supply, operating system, RAM, CD-ROM, graphic adapter, monitor, mouse, keyboard, digital audio system, videodisc player, and network.

- AGR 003 Digital Audio - Recommends guidelines that promote the interoperability of digital audio. Interoperability means the ability of courseware with audio to playback on different PCs with different audio cards in them. It also means the ability of a single PC to playback courses with audio from different vendors.

- AGR 004 Operating/Windowing System - Provides a formal recommendation to the aviation industry for an operating and windowing system used for delivery of CBT. It contains the results of a survey of the major operating and windowing systems conducted by the AICC.

- AGR 005 CBT Peripheral Devices - Recommends guidelines that promote the interoperability of the following peripheral devices: video overlay card, videodisk player, and XY input device (such as a touch screen, mouse, or trackball), and part task trainers.

- AGR 006 Computer-Managed Instruction - Recommends guidelines that promote the interoperability of CMI systems (on local file systems). Interoperability means the ability of a given CMI system to manage CBT lessons from different origins. It also includes the ability for a given CBT lesson to exchange data with different CMI systems.

- AGR 007 Courseware Interchange - Recommends guidelines for the interchange of the elements that occur in CBT courseware including text, graphics, motion (frame-based), audio, and logic. These guidelines encompass the major data components of CBT courseware, and standard data formats for those components.

- AGR 008 Digital Video - Recommends guidelines for the creation, distribution, and use of digital video in CBT courseware.

- AGR 009 - Icon Standards: User Interface - Recommends guidelines for the functions of the student/user interface and their associated graphic representation in CBT courseware.

- AGR 010 - Web-Based Computer-Managed Instruction - Recommends guidelines that promote the interoperability of web-based CMI systems.

6. Glossary of Terms

ADL – Advanced Distributed Learning
AEC – Advanced Electronic Classrooms
AICC – Aviation Industry CBT (Computer-Based Training) Committee (AICC)
ANSI – American National Standards Institute
API – application programmer interface
avatar – an icon or actor used to representation the user inside the virtual world
AVI – Audio Video Interleave file format
C4 – Command, Control, Communications, & Computers
CBT – Computer-based Training
CD – compact disc
CFFC – Commander, Fleet Forces Command
CIO – Chief Information Officer
CMI – Computer Managed Instruction
CMS – Course Management System
codec – compression/decompression software
COE – Common Operating Environment
COI – Course of Instruction
DIS – Distributed Interactive Simulation
ebXML - Electronic Business using eXtensible Markup Language
H-Anim – Human Animation standard of Web 3D Consortium
HLA – High Level Architecture
HPSM – Human Performance System Model
IEEE – Institute of Electrical and Electronics Engineers
IGES – Initial Graphics Exchange Specification
ILE – Integrated Learning Environment
ISO – International Organization for Standardization
JAVA – a mobile code programming language
KIF – Knowledge Interchange Format
LAN – Local Area Network
LCMS – Learning Content Management
LMS – Learning Management System
mobile code – software obtained from remote systems, transferred across a network, and then downloaded and executed on a local system without explicit installation or execution by the recipient
MPEG – a series of audio/visual data standards from the Moving Picture Experts Group of ISO
NETC – Naval Education and Training Command
NIST – National Institute of Standards and Technology
OMG – Object Management Group
OMT – Object Modeling Template
OPNAV – Chief of Naval Operations
OWL – Web Ontology Language
PC – Personal Computer
PDA – Personal Digital Assistant, a handheld computer
RAM – Random Access Memory

RDF – Resource Description Framework
RLO – reusable learning object
RTI – Run-Time Infrastructure
scaffolding – structures built into the scenario, simulated world, roles, and tasks of a simulation-based learning application that help the student to achieve the desired learning experience.
SCO – Shareable Content Object
SCORM – Shareable Content Object Reference Model
SGML – Standard Generalized Markup Language
STEP – STandard for the Exchange of Product model data
VRML – Virtual Reality Modeling Language
W3C – World Wide Web Consortium
WAN – Wide Area Network
XML – Extensible Markup Language

7. References

[Alexander 2003] Alexander, Thor (ed.), Massively Multiplayer Game Development, Charles River Media, Hingham, MA, 2003.

[Banks 1998] Banks, Jerry (ed.), Handbook of Simulation: Principles, Methodology, Advances, Applications, and Practice, John Wiley and Sons, New York, NY, 1998.

[Bloom 1964] Bloom, B. S., Mesia, B. B., & Krathwohl, D. R., Taxonomy of Educational Objectives (two vols: The Affective Domain & The Cognitive Domain), New York, 1964

[Brandon Hall 2003] E-Learning Simulations: Tools and Services for Creating Software, Business, and Technical Skills Simulations by the staff of brandon-hall.com, 2003.

[Goldfarb 2000] Goldfarb, Charles F. and Prescod, Paul, The XML Handbook, Prentice Hall, Upper Saddle River, NJ, 2000.

[HPC 2003] Human Performance Center, "Human Performance Professional Working Guidelines," October 2003.

[Kuhl 1999] Kuhl, Dr. Frederick, Weatherly, Dr. Richard, and Dahmann, Dr. Judith, Creating Computer Simulations: An Introduction to the High Level Architecture, Prentice Hall, Upper Saddle River, NJ, 1999.

[DODJS 2003] Department of Defense Joint Staff, "Defense-in-Depth: Information Assurance (IA) and Computer Network Defense (CND)," 25 March 2003.

[Libicki 1995] Libicki, Martin C., Information Technology Standards: Quest for the Common Byte, Digital Press, Boston, 1995.

[Murray 1996] Murray, James D. and VanRyper, William, Encyclopedia of Graphics File Formats/Book and Cd Rom, O'Reilly & Associates, Bonn, 1996.

[NASA 2003] Moran, Patrick J., "An Open Source Option for NASA Software," NAS Technical Report NAS-03-009, NASA Ames Research Center, Moffett Field CA, April 2003.

[NPDC 2003] Naval Personnel Development Command, "Integrated Learning Environment Guidance Content Development Regarding Navy Enterprise Standards and Certification," 2003.

[OSI 2004] Open Source Initiative, http://www.opensource.org/.

[Shaw 1996] Shaw, Mary and Garlan, David, Software Architecture: Perspectives on an Emerging Discipline, Prentice-Hall: Saddle River, NJ, 1996.

[Sommerville 2000] Sommerville, Ian, Software Engineering, Addison-Wesley: Boston, MA, 2000.

[Spivak 2001] Spivak, Steven and Brenner, F. Cecil, Standardization Essentials: Principles and Practice, Marcel Dekker, New York, 2001.

[Winston 1992] Winston, Patrick, Artificial Intelligence, Addison-Wesley: Boston, MA, 1992.

Appendix A. NMCI Gold Disk Contents

o

SERVICE	SOFTWARE DESCRIPTION (MINIMUM VERSION)	VENDOR
Operating System	MS Windows 2000 Build 2195 SP1	Microsoft
Office Suite	MS Office Pro 2000 SR-1a	Microsoft
Email Client	MS Outlook 2000	Microsoft
Internet Browser	MS Internet Explorer 5.5 SP-1 128bit	Microsoft
Virus Protection	Norton A/V Corp Edition v7.5	Symantec
PDF Viewer	Acrobat Reader v.5.0	Adobe
Terminal Emulator - Host (TN3270, VT100, X-Terminal)	Reflection 8.0.5	WRQ
Compression Tool	Winzip v.8	Winzip
Collaboration Tool	Net Meeting v3.01 (4.4.3385)	Microsoft
MultiMedia	RealPlayer 8	RealNetworks
MultiMedia	Windows Media Player v7.0.0.1956	Microsoft
Internet Browser	Communicator 4.76	Netscape
Plug-ins		
Web Controls	MacroMedia Shockwave v 8.0	MacroMedia
Web Controls	Flash Player 5.0	MacroMedia
Web Controls	Apple Quicktime Movie and Audio Viewer v4.12	Apple
Web Controls	IPIX v6,2,0,5	Internet Pictures
Security Apps		
Security	Intruder Alert v3.5	Axent
Security	ESM v5.1	Axent
Agents		
Software Management	Radia Client Connect	Novadigm
Inventory, Remote control	Tivoli TMA 3.7	IBM/Tivoli
Remote Connectivity (Notebooks)		
Dial-up connectivity	PAL	MCI/Worldcom
VPN	VPN Client	Alcatel

Microsoft Office Professional 2000 includes:

- Word
- Excel
- Outlook
- Publisher
- Small Business Tools
- Access
- PowerPoint

Reference: https://nmci.navair.navy.mil/war_room/Talking_Points_-_App_Mapping_-_02-14.DOC

Appendix B. Mobile Code Technology

The following table provides examples of mobile code technologies for illustration purposes, the technologies are not necessarily approved for use on Navy PC simulation-based learning platforms.

Table B-1 Examples of Mobile Code Technologies

ActiveX	VBScript	Active Server Pages (ASP)
Windows Scripting Host (WSH) –when used to execute mobile code	Portable Document Format	Cold Fusion Markup Language (CFML)
UNIX Shell Scripts –when used to execute mobile code	Shockwave/Flash	Hypertext Preprocessor (PHP)
Disk Operating System (DOS) Batch Scripts –when used to execute mobile code	Extensible Markup Language (XML)	Server Sided Include (SSI)
Java Applets and other Java Mobile Code	Synchronized Multimedia Integration Language (SMIL)	Server-sided JavaScript
Visual Basic for Applications (VBA)	Quicktime	Server-sided LotusScript
LotusScript	Virtual Reality Markup Language (VRML) – exclusive of any associated Java applets or JavaScript scripts.	Java Remote Method Invocation (RMI)
PerfectScript	Java servlets	Java Network Device Connection (JINI)
Postscript	Java Server Pages	
JavaScript (including Jscript and ECMAScript variants	Common Gateway Interface (CGI)	

Appendix C. Configuration of the Integrated Learning Environment (ILE)

The following table identifies the current configuration of the ILE environment in terms of architecture modules, associate systems/versions, and locations where the modules are installed.

Table C-1 Configuration of the Integrated Learning Environment

Architectural Module	Software System/Version	Locations
Learning Content Management System (LCMS)	Outstart Evolution-Version 2.0	NETC, Pensacola, FL
Learning Management System (LMS)	Thinq-Version X.X	NETC, Pensacola, FL
Student Learning Platform PC (SLP-PC)[1]	*Hardware, OS, to be determined.*	Various
Student Learning Platform PDA (SLP-PDA)	*Hardware, OS, to be determined.*	Various
Web Browser	Internet Explorer Netscape Navigator	Various
Web Browser Plug-Ins[2]	Real Player-Version 8.x Windows Media Player-Version 7.0 Adobe Reader-Version 4.05 Macromedia Shockwave-Version 8.0 Macromedia Authorware Web Player Version 5.1 Macromedia Flash Player, Version 5 Apple QuickTime Movie and Audio Viewer-Version 4.0 IPIX-Version 6.2	Various
Architectural Module	**Software System/Version**	**Locations**

[1] *Student Learning Platform-PC* - The target common operating environment (COE) for running PC simulation-based learning systems is the Navy Information Technology for the 21st Century (IT-21) specifications. The IT-21 platform minimally meets and usually exceeds the industry standard Multimedia PC Council recommendations. As such, PC simulations should not require any special considerations from the aspect of hardware design and integration. The Navy is transitioning to the Navy Marine Corps Intranet (NMCI). NMCI workstation configurations far exceed that of IT-21. PC simulation vendors should ensure that their content runs on the NMCI-configured SLPs. NMCI computer security policies allow a system administrator to install applications that write to their own areas of the disk during the installation process. Applications must refrain from writing to restricted portions of the registry or other non-authorized areas of the disk at runtime. See the Integrated Learning Environment (ILE) and IT-21 documents for more current and detailed configuration information.

[2] *Browser Plug-Ins* - Newer versions of the listed plug-ins are under consideration. Review the appropriate ILE (NMCI) documents for the latest versions of plug-ins supported. PC Simulation vendors wishing to introduce alternative plug-ins for their products will need to submit the products for testing. See Appendix A and URL reference for further guidance.

Course Authoring System (CAS)	Outstart Evolution-Version 2.0	Various
Distributed Simulation Component (DSC)	*To be determined.*	Various
Stand-Alone Simulator (SAS)	*To be determined.*	Various
Courseware Testing System (CTS)	*To be determined.*	Various
Course Management System (CMS)	*To be determined.*	Various

Appendix D. Selected Standards

This section identifies the standards that have currently been selected for implementation of Navy PC simulation-based learning systems.

Architectural Interface	Applicable Standards and/or Interface Specifications
Reusable Learning Object Data Formats	ADL Sharable Content Object Reference Model (SCORM) + extensions
Learning Content Management System (LCMS) to Learning Management System (LMS) Application Programmer Interface (API)	*To be specified*
Web Browser (WB) and Web Browser Plug-In (WBPI) Courseware to Learning Management System (LMS) Interface	*To be specified*
Student Learning Platform-PC (SLP-PC) to Student Learning Platform-Personal Digital Assistant (SLP-PDA) Interface	*To be specified*
Distributed Simulation Component (DSC) Interface	*To be specified*
Courseware to Learning Management System (LMS) Interface	*To be specified*

Appendix E. Brief Overview of a Game Engine Architecture

This appendix provides a brief overview of a generic video game engine architecture. The architecture divides the engine into a set of component modules and interfaces. Component modules may be replaced with software from different developers, as long as the interfaces conform to the specifications defined by the architecture.

The major component elements of the game engine architecture are:

1) Supervisory controller
2) User input module
3) Script interpreter
4) Data management module
5) Data import/export module
6) Game logic module
7) Graphics module
8) Sound module
9) Animation module
10) Physics module
11) Artificial intelligence module
12) Multi-player operations and server support module
13) Learning system support module
14) Game level editor

Each of the major component modules may be further decomposed into sub-modules. A description of all possible sub-modules and interface specifications between modules is beyond the scope of this document. Other development software and systems that may be used to create data for the game level editor are not addressed, e.g., 3D Studio Max to create graphical objects, Adobe Photoshop to create textures, or motion capture systems to animate characters. Also the implementation of a server that interacts with the game engine's server support modules is not discussed.

Although there is a close relationship to characteristics that were defined to assess game engine capabilities, there is not a one-to-one mapping between characteristics and modules in the architecture. The assessment mostly deals with externally observable features of game engines. The preliminary architecture deals with the internal structure of the engine. Furthermore, some characteristics were combined for purposes of simplifying the assessment. In some cases these characteristics need to be broken out to better understand the architecture. For example, where sound and graphics capabilities may be treated together in assessing most game engines, functionally they will be separate modules in the architecture. Also the functions of the supervisory controller, user input management, script interpretation, game control, and data management were combined under the characteristic called "game management."

On the other hand, some capabilities that were separated in the assessment have been combined in the architecture. For example, in our architecture the functionality associated with software

distribution will ultimately be implemented on the client side in the learning system support module. Distribution and security requirements may also affect the coding practices used in the implementation of all other modules.

The functions of each of the modules are briefly described below.

1) *Supervisory controller* –This module is the core of the game engine. It is responsible for execution control and supervisory capabilities that are built into the software. It can be thought of as the control logic, the program loop, or the operating system that runs the game. It is responsible for coordinating the other modules in the system. It is in part what distinguishes a game engine from a library of subroutines. It manages the initialization and shutdown of the software, loading of scripts, setup of services provided by other modules, execution of game logic, display of multiple output windows, etc. It manages the threaded execution of scripts that have been loaded into the game logic module. It provides interfaces to a game level editor that allows users/developers to create new game experiences at a higher level using a scripting language, thus avoiding low-level programming.

   The module is responsible for scheduling or sequencing module execution to ensure that scene graphs get updated and graphics objects are rendered smoothly. It also ensures that updates to data, performance of physics calculations, and other time critical computations occur in a timely manner. As such, it is responsible for the system clock and calendar, the management of time, management of triggers, and the scheduling of events. It provides functions for saving and loading game levels and game state, pausing execution, playing back and rolling back game execution. It may manage the messaging scheme by which communications occur between internal modules. It also initiates data updates with other executing game engines (via the multi-player operations and server support module).

2) *User input module* – The user input module is responsible for interfacing with the physical input devices and the associated device drivers connected to the user's computer. The module translates interactions with peripheral input devices into meaningful messages in the context of the game world. Input devices typically include the keyboard, mouse, joystick, or other special purpose input peripherals (e.g., game controllers, steering wheels, tablets).

   The output of an artificial intelligence speech recognition sub-module may also be directed to the user input module, as if it were another physical input device. If voice commands are accepted as inputs, the audio card connected to the sound module first digitizes the speech sounds. The digitized sounds are translated into phonemes and assembled into words by an AI speech recognition sub-module. The words are parsed into phrases that are translated into messages passed from the user input module.

   The user input module is driven by the input expectations of the game logic. As such, the game logic module must tell the user input module what inputs it is currently expecting or willing to accept, which modules to notify when an input is received, and what messages to send to those modules based upon the user's keyboard, mouse, voice command, or other actions.

3) *Script interpreter* – This module is responsible for the execution of high-level programs called "scripts." Scripts define the evolution of the game, the interactions between players and game objects, the behaviors of game objects, and invoke functions provided by other modules in the game engine. Scripts are used in two places within the engine: 1) the game logic module, and 2) game objects in the data management module.

   The scripting language is used to define the execution of the game and the behaviors of the objects. Scripting languages that have been used in previous game engines include: Python, Ruby, and Lua. The script interpreter will likely be based on variations of one or more of these scripting languages. Although access to script interpreters, may be allowed in commercial games that allow user level modifications, or "mods," this will probably not be allowed for Navy students both for security and system integrity reasons.

   The scripting language helps improve the reliability of the overall system and restricts the ability of developers to introduce security holes into the system. The use of scripts eliminates the need for all developers to create and compile low-level code that is typically written in C or C++. Content developers will typically use a script interpreter to interactively test scripts during the development process. An existing script compiler will also be integrated and configured to translate scripts into machine code, thus accelerating the execution of scripts. Compiled scripts will also enhance the security of the system by increasing the difficulty of hacking or tampering with scripts.

4) *Data management module* – This module is responsible for managing the internal database of data objects that define the game. All active data objects are stored in the data management module. Scripts may appear on the attributes of objects stored in this module.

   The module provides an organizational structure for data objects, indexes objects, and manages memory. It creates instances of objects, handles loading of object data via the import/export module, updates to data attribute on objects, viewing of objects for developers, and performs garbage collection as object data storage is no longer needed.

5) *Data import/export module* – This module is responsible for translating the data contained in external files into internal engine formats, and conversely translating internal engine data into external file formats. Parser sub-modules will be needed to read external files in the various formats. Generator sub-modules will be needed to output data from internal structures to output file formats.

   Relevant data formats include those for graphics files, video, audio, game levels, game state, character models, motion capture, RLOs, XML objects, etc. Graphics files include those to describe three-dimensional objects, textures, and motion capture. Most engines provide some support for 3D Studio Max and Maya formats. Other graphics file formats include JPEG, TIFF, DXF, and BMP. Motion capture files will be needed to support character animation including: BIP, CSM, and BVH files. Multi-media and sound files also need to be supported, such as MPEG, WAV, MIDI, and AVI files. Learning data based upon Reusable Learning Objects (RLOs) that are SCORM-compliant and implemented using the Extensible

Markup Language (XML) must be supported. SEDRIS capabilities will be needed to import and export environmental data.

Data import and export operations are invoked from the scripting language. The data management module handles imported data. Exported data is also obtained from the data management module.

6) *Game Logic Module* – This module is responsible for the execution of scripts that are not defined on attributes of objects in the data management module. It may be thought of as the program memory area that is accessible to learning content developers. Multiple scripts may be loaded into the game logic module for concurrent or sequential execution. The supervisory controller is responsible for coordinating the timely execution of scripts in this module.

   The module is where the content developers will load the scripts that define the overall action of the game or simulation. A process table will allow the developer to define how and when a particular script should run, e.g., priority level. The execution of scripts in the game logic module may cause updates to objects in the data management module or the execution of scripts on objects in the data management module.

7) *Graphics module* – This module is responsible for generation of two- and three- dimensional graphics. Graphics includes the capability to generate images objects and scenes, map textures onto objects, provide various forms of lighting, cast shadows, model fog and smoke, and provide cameras (viewing controls), among other features. Realism is further enhanced by the photographic quality, level of detail, the seamless joining of texture maps on surfaces, forms of randomness in images (dirt, rust, etc.), and other factors that typically distinguish a photograph from a computer-generated image. Graphics object libraries and procedurally generated graphical objects (e.g., leaves on trees, large numbers of buildings in a city) can further enhance realism.

   The graphics module interfaces to the computer's graphics card using OpenGL or Microsoft DirectX graphics software libraries. The data passed to the card includes polygons, vertices, colors, textures, etc.

8) *Sound module* – This module is responsible for all audio input and output. It generates audio output and handles audio input through the audio hardware and associated device drivers. The module simultaneously plays sound sequences loaded from multiple files encoded in standard formats. Some examples of sound file formats include: WAV, MIDI, MP3, and OGG.

   The sound module creates special sound effects based on audio programming, generates speech audio from text input, and handles various sound effects. Sound effects include three-dimensional location of sounds, ambient noise, handling of reverberation, pitch changes, fades, distortions, and echoes that are associated with the behavior of sound in different spaces, movement of sound sources, etc.

It handles the timing of audio events. Audio events are triggered either by scripts on game objects or by scripts in the game logic module. The scripting language invokes the sound module by subroutine call. The subroutine calls available through the scripting language will be addressed in a separate specification.

The module also handles audio capture, i.e., the digitization of input sound from a player's microphone. Digitized audio is then either saved to a file, passed to AI modules for speech recognition, or in the case of multi-player games, to other players using Voice-over-Internet.

The sound module is implemented in C and/or C++ using the audio extensions to OpenGL (OpenAL) and/or DirectX (DirectSound and others). The policy on use of OpenGL versus DirectX is to be determined. In any case, the internal programming of the sound module will not be accessible to content developers. The architecture of the sound module will be further decomposed into sub-modules.

9) *Animation module* – The module handles the animation, i.e., the coordinated movement, shape changes, morphs, etc., of graphical objects over time. Objects that may be animated are called "actors." Actors include characters, vehicles, multi-body and movable objects. Sub-modules may be defined to implement specific motions associated with different types of actors.

   The module supports specification of range of motion for body joints, key frame animation, and the implementation of intelligent behavior. Intelligent behavior refers to the ability to perform complex sequence of activities that have been built up from basic motions using such techniques as finite state machines.

   The module handles the smooth interpolation of body joint regions when actors are in motion, blending of different motion sequences, footstep locomotion, motion capture capabilities (using performers outfitted with special hardware), modeling of the movement of cloth and hair, physics of character interaction (bumping into each other or falling down), manipulation of various props and objects. If motion capture is supported, any of several motion capture file formats may be used (e.g., BIP, CSM, and BVH files).

10) *Physics module* - The physics module is decomposed into a collection of sub-modules. The sub-modules are responsible for performing the calculations that enable the realistic display of collisions, trajectories of projectiles, falling objects, cornering of vehicles, floating and sinking objects (e.g., vessels), movements of cloth and hair, etc. Sub-modules may also be used to model other physical, chemical processes and phenomena, such as the environment (weather, sea conditions such as waves and swells), fires, the flow of liquids and gases, burn rates, behavior of electromagnetic and sound waves, etc. within the scope of the physics module. Proximity sensors are required by the physics module to implement interactions between objects.

    Basic principles that are relevant to the implementation of physics modules include mass, center of mass, volume, Newton's laws, inertia, units of measure, geometry and vectors. Other concepts include kinematics of objects, linkages of objects, their linear and angular

velocity, forces and torques acting on the objects, acceleration, and momentum in two and three dimensions. The physics module requires functions for performing physics calculations as well as data attributes on the objects to support those calculations. Soft body or rigid body mechanics may be used to model the behavior of objects in collision. Soft body mechanics will provide more realistic behavior with greater computational costs.

11) *Artificial intelligence module* – Artificial intelligence (AI) may provide a number of different types of processing capabilities that contribute to the overall realism in the game environment. AI functions may include speech recognition, natural language processing, path planning (e.g., autonomous movements of characters, vehicles, watercrafts, aircraft, etc.), development of strategies for non-player characters, coordinated behaviors, flocking, smart terrain, automatic solution of lower level behaviors for player characters, character motion, learning, etc.

    AI techniques used to implement sub-modules may include genetic algorithms, neural networks, randomness and statistical behaviors, rule-based systems, decision trees, goal-directed behavior, problem solving systems, fuzzy logic, and blackboard systems. AI sub-modules will be invoked using the scripting language from either the game logic module or the objects managed by the data management module. The results of AI processing will be returned to game logic or game objects that invoked the processing.

12) *Multi-player operations and server support module* – This module implements distributed, multi-player simulation and gaming capabilities using mechanisms such as the DoD's High Level Architecture (HLA) Run-Time Infrastructure (RTI), Distributed Interactive Simulation (DIS), Simulator Networking (SIMNET), or Massively Multi-Player (MMP) games. It also provides Voice Over Internet Protocol (VoIP) mechanisms for carrying out voice communications between players and instructors.

    The server support module is responsible for sending and receiving data that is needed by the server's clients or other student's interacting in the same distributed simulation environment. Data includes digitized voice objects, streaming audio, and or video. The module is also responsible for handling time synchronization of the simulation with the server and/or other client applications. The scripting language will be used to identify data distribution requirements. Depending on the mechanism used to support distributed operations; a server may be required on another computer platform to support the student's client computers.

13) *Learning system support module* – This module is one of the major differences between this architecture and that associated with a traditional game engine. Interfaces to LCMS and LMS software do not exist in traditional video game applications that have been developed for the entertainment market. The module provides functions to enable the use of the game engine in an instructional setting. Due to the large numbers of potential users of Navy simulation-based learning applications, video game-based learning application and courseware must be distributed over the Internet as web page downloads. The use of CD-ROMs or physical media of any sort must be avoided, if at all possible. The game engine and the courseware must have a small footprint on the client machine (i.e., use a minimal amount of disk space) and be capable of being downloaded incrementally as needed. This

module is responsible for managing incremental downloads of game engine software and learning content. Long download times during course execution are undesirable. Security constraints associated with the Navy Marine Corps Intranet (NMCI) environment must also be addressed.

The learning system support module must interface to a Learning Content Management System (LCMS) and a Learning Management System (LMS). It will allow the student to access course content, automatically retrieve game/learning objects, manage evolution of instruction, and track the student's progress. Content is downloaded from an LCMS repository. Content is organized into game engine software, application objects, supporting traditional instructional structures (courses, lessons, etc.). The LMS interface is used to construct and vary lesson scenarios for repeat execution, scoring progress against learning objectives, tracking student progress, providing on-line help associated with instructional programs, etc.

14) *Game level editor* - The game level editor (and associated external development tools such as 3D Studio Max, Adobe Photoshop, or Maya) are the primary way that content will be developed. The functionality of the editor allows developers to work at a high level when creating learning content. The editor contains sub-modules for creating 3D environments, player and non-player characters, and supporting objects. It also supports character animation through the development of character models (possibly using other character modeling software tools), definition of bones (controls for body segments). It is used to create scripts for the game logic module or game objects.

    It must provide a seamless interface to the game engine itself for testing and debugging objects and scripts. It must allow the users to create behaviors on these characters and objects, define game levels, establish lesson plans, scoring schemes, etc. It also must provide capabilities for defining download data sets, i.e., packages of objects that are downloaded as a set when a student is interacting with the learning environment.

    The editor must be well documented and easy to use. It should provide on-line help and a rich set of examples to help users get started.