Potential Technologies for Engineering Network-Centric Simulations

Osman Balci

Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA
balci@vt.edu

Ernest H. Page

The MITRE Corporation
7515 Colshire Drive
McLean, VA 22102, USA
epage@mitre.org

Keywords: Distributed M&S, High Level Architecture, Java EE, .NET framework, web-based M&S

Abstract

The paradigm under which we engineer and use software has changed from a product-oriented view to a service-oriented view in recent years. Technologies have emerged to enable the development and deployment of software applications to run on server computers for use by many people over a network. This paper describes such technologies for modeling and simulation (M&S) applications. Namely, Java platform, Enterprise Edition, Microsoft platform, .NET Framework, and High Level Architecture technologies are described for building network-centric M&S applications.

1. INTRODUCTION

A major paradigm shift has occurred in recent years in the way we engineer and use software. Traditionally, we have engineered Software as a Product (SaaP), which was sold in a shrink-wrapped box as a product, purchased and installed on a single computer for use by a single user. Under the new paradigm, we engineer a software application and run it on a server computer for use by many people over a network such as Internet, virtual private network, or wireless network. This new paradigm is called Software as a Service (SaaS), which is sometimes referred to as Cloud Computing, where cloud is a metaphor for network. Another term commonly used for the new SaaS paradigm is network-centric software engineering, which we adopt in this paper.

A network-centric Modeling and Simulation (M&S) application is the one that runs on one or more server computers and used by many people over a network. Many architectures can be employed for engineering a network-centric M&S application such as client-server architecture, distributed objects architecture, high level architecture (HLA) [IEEE 2000], and service-oriented architecture (SOA).

A network-centric M&S application can be created for problem solving or training purposes. It is especially needed for training geographically dispersed people over a local or wide area network. For example, we can develop a network-centric M&S application to train first responders, officials, decision makers, and public as part of an emergency response management plan for a city. A user can use the M&S application under a variety of simulated scenarios for the purpose of learning what to do in case of an emergency as specified in the plan.

The purpose of this paper is to describe some technologies for engineering network-centric M&S applications. Two industry-standard platforms have emerged as a result of the paradigm shift from SaaP to SaaS: Java platform, Enterprise Edition (Java EE) and the Microsoft platform, .NET Framework. Java EE is described in Section 2. The .NET Framework is presented in Section 3. Section 4 describes the classical HLA for building network-centric simulations. Concluding remarks are given in Section 5.

2. JAVA PLATFORM, ENTERPRISE EDITION

Java platform, Enterprise Edition (Java EE) [Sun 2009a] is used for engineering network-centric software systems. It can also be used for developing network-centric modeling and simulation (M&S) applications. Java EE consists of five layers as depicted in Figure 1 with circled numbers.

Layer 1 represents a Relational DataBase Management System (RDBMS) such as Oracle database [Oracle 2009], IBM DB2 [IBM 2009a], Microsoft SQL Server [Microsoft 2009b], and PostgreSQL [PostgreSQL 2009]. PostgreSQL is a free open source RDBMS product.

Layer 2 represents the data mapping achieved by using typically Entity Enterprise JavaBeans (EJBs). Entity EJBs are created to hold the RDBMS data in memory for the purpose of improving the execution efficiency. With data mapping, read-only data can be provided from the main memory directly by eliminating the need to access the database and hence, resulting in better performance.

Layers 3 and 4 are part of the Java EE Application Server software product. Many Application Server products are available for the Java platform including Apache Geronimo, WebLogic Server, IBM WebSphere Application Server, JBossAS, Oracle Application Server, and Sun

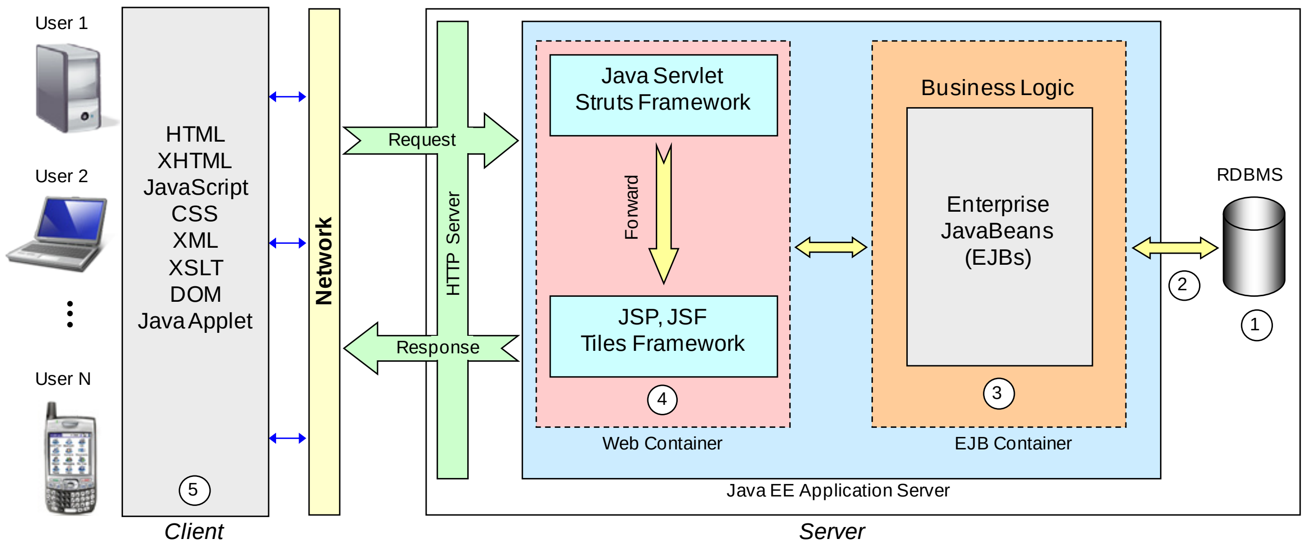

Figure 1. Java Platform, Enterprise Edition (Java EE)

GlassFish Enterprise Server. Sun [2009b] provides a list of Java EE version 5 compatible application servers.

A Java EE compatible application server consists of two containers: EJB and Web. Containers are basically run-time environments that provide specific services to applications. EJB Container provides automated support for transaction and state management of EJB components, as well as look up, security services, concurrency, and memory management. The EJB Container is designated as Layer 3. Web Container provides run-time support to clients by processing requests with Java Servlets, Java Server Pages (JSP), and Java Server Faces (JSF), and returning results from the EJB components to the clients. The Web Container is designated as Layer 4.

Layer 3 represents the application logic or business logic programmed by using EJBs. EJB Container manages and controls the execution of all EJBs. A simulation model and its execution are provided in this layer. Programming of the simulation model can be accomplished using more than 3500 classes provided as part of the Java EE Integrated Development Environment (IDE). Documentation of all classes is made available by Sun [2009c].

Each application server has an associated IDE. For example, IBM's IDE is called IBM Rational Application Developer (RAD) for WebSphere Software [IBM 2009b]. RAD can be used to develop a network-centric simulation model, which can be deployed to run on a server computer with IBM WebSphere Application Server.

Layer 4 provides two major functions: (1) controlling and mediating between the client and the application running on the server computer. Java servlets are typically used for this function. Struts Framework is commonly used as the best practice to implement this function. (2) server-side preparation of the presentation to be sent to the client. JSP is used for this function together with JSF. Tiles Framework is commonly used as the best practice to implement this function.

Layer 5 represents the client computer with a web browser. The user uses the web browser to interact with the software/M&S application running on server computer over a network such as Internet, Wireless Network, or Virtual Private Network. HyperText Transfer Protocol (HTTP) is used for the communication between the client and server computer over a network. HTTP is a Request/Response-oriented protocol, which runs on top of the Transmission Control Protocol / Internet Protocol (TCP/IP).

Layer 5 provides the user interface of a network-centric software/M&S application with technologies such as Cascading Style Sheets (CSS), Document Object Model (DOM), HyperText Markup Language (HTML), Extensible HTML (XHTML), Extensible Markup Language (XML), Extensible Stylesheet Language Transformation (XSLT), JavaScript, and JavaApplet.

Myers and Balci [2009] demonstrate the use of the Scalable Vector Graphics (SVG) and XML for creating a Java EE-based architecture for network-centric visual simulations as depicted in Figure 2. The architecture decomposes into two basic operations: composing a simulation model and executing a simulation model. Each operation represents a data flow through a series of components into (system input) or from (system output) the data source layer. During each data flow, a single architecture component in each layer takes responsibility for handling the communication to and from the layer. The placement of the individual components in the architecture represents how the communication is handled by each layer. Components located in the upper portion of the architectural diagram represent input communication handlers while

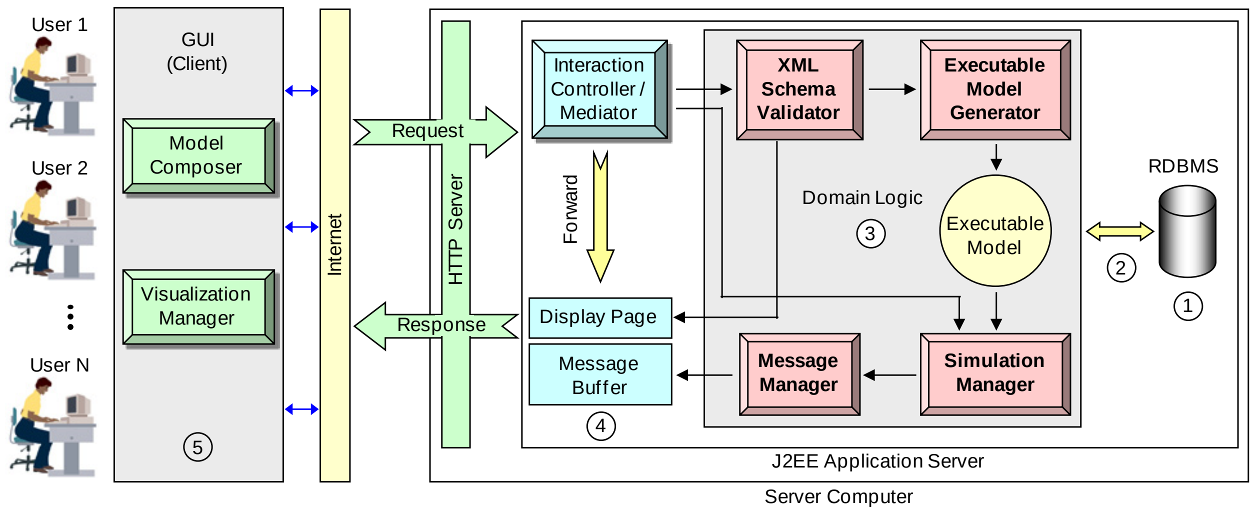

Figure 2. Java EE-based Client-Server Architecture for Visual Simulation [Myers and Balci 2009]

components in the bottom portion of the architectural diagram represent output communication handlers.

3. MICROSOFT PLATFORM, .NET FRAMEWORK

Microsoft platform, .NET framework [Microsoft 2009a] consists of five layers as depicted in Figure 3 with circled numbers. It can be used for engineering network-centric M&S applications.

For Layer 1, any RDBMS can be used; however, Microsoft SQL Server [Microsoft 2009b] provides a better integration.

Layer 2 represents the data mapping achieved by using ActiveX Data Objects (ADO). ADO is a language-neutral object model that exposes RDBMS data sources to ADO. With data mapping, read-only data can be provided from the main memory directly by eliminating the need to access the database and hence, resulting in better performance.

The .NET application server is embedded within the Windows Server 2008 operating system. It includes the Common Language Runtime (CLR), which is a runtime engine that (a) manages the execution of the .NET code, and (b) provides features such as memory management, thread management, object type safety, and security. CLR can manage the execution of any code that can be translated into the Intermediate Language (IL) representation. IL is a hardware independent machine language representation of a source code, which can be just-in-time compiled for native execution on a specific hardware. IL translators are available for source code written in many languages including C#, Visual Basic, C++, and Java.

In concept, IL corresponds to Java byte code under the Java technology. However, the .NET framework provides the byte code-like IL representation not only for Java but for many programming languages. On the other hand, EJBs are components that can be reused by all Java platform developers. CLR-managed components can be reused only under a particular programming language, which hinders the ability to reuse.

An M&S application can be implemented using a programming language for which an IL translator is available (e.g., C++, C#, Java) so that the M&S application execution can be managed by CLR. This is accomplished in Layer 3 of the architecture. An M&S application can be developed to run under the .NET engine by using the Visual Studio 2008 Integrated Development Environment (IDE), which is the natural IDE for .NET application development. The .NET framework class library provides a collection of reusable classes, interfaces, enumerations, and structures for use by a managed M&S application code.

Layer 4 provides two major functions: (1) controlling and mediating between the client and the M&S application running on the server computer. Internet Information Services (IIS) is typically used for this function. (2) server-side preparation of the presentation to be sent to the client. This is accomplished by using ASP.NET, which is the next generation of the Active Server Page (ASP) technology.

Layer 5 provides the user interface of a network-centric M&S application with technologies listed in Figure 3. Most of the technologies provided in this layer are the same as those used for Java EE.

The M&S application running in Layer 3 under either the Java or Microsoft platform can call upon web services provided by other applications. For example, Sabah and Balci [2005] provide Random Variate Generation (RVG) web services. A network-centric simulation model can call the RVG web services over the network and use the random variates generated from more than two dozen probability distributions by the RVG web services. Thus, an M&S application can be built under the service-oriented architecture (SOA) using Java or Microsoft platforms.

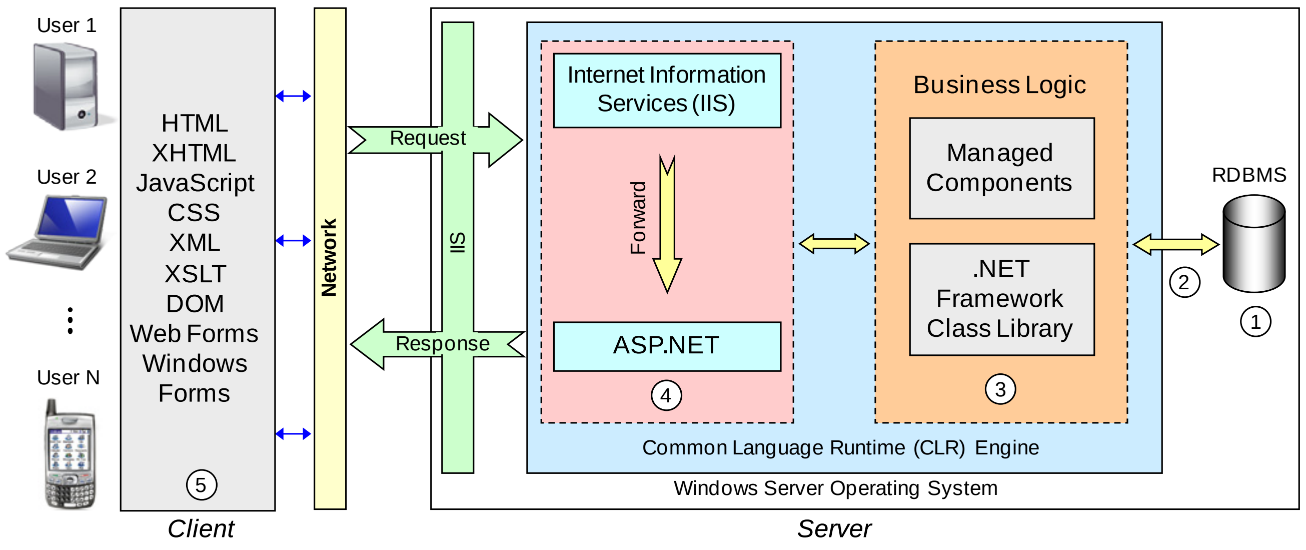

Figure 3. Microsoft Platform, .NET Framework

4. HIGH LEVEL ARCHITECTURE

The High Level Architecture (HLA) for M&S was developed through the U.S. Department of Defense (DoD) during the mid-1990s, as part of its ongoing pursuit of standards and technologies to support the interoperability of DoD simulations. Scores of articles about the HLA and its predecessors have appeared during the past decade and a half. The uninitiated reader should consult the book by Kuhl, Weatherly, and Dahmann [1999] for a broad introduction to HLA and its earliest implementations. The proceedings of the semiannual Simulation Interoperability Workshop (SIW) provide details on current research and development within the HLA and related technologies [SISO 2009]. The book by Fujimoto [2000] addresses HLA within the context of parallel and distributed simulation frameworks.

As noted by Kuhl, Weatherly and Dahmann [1999], the design of the HLA is based on several premises:

- No single, monolithic simulation can satisfy the needs of all users. Users differ in their interests and requirements for fidelity and detail.
- Simulation developers vary in their knowledge of domains to be simulated. No single set of developers is expert across all details even in one domain.
- No one can anticipate all the uses of simulation and all the ways simulations could be usefully combined. Even if a developer were able to satisfy a comprehensive set of requirements in a domain of application (by building the monolithic simulation), the developer could not anticipate the requirements for such a system. The world changes continually; nobody knows all the future uses for simulation, even in one domain.
- Future technology and tools must be incorporated. If developers were able to build the comprehensive simulation, and their crystal ball enabled them to anticipate requirements, computer technology would change underneath them. If their crystal ball could reveal the future of computer technology, they would still face the problem of incorporating it as their customers demanded to benefit from its advances.

HLA consists of three components: (1) a model definition and specification formalism called the *Object Model Template* (OMT) specifying what information is communicated between simulations, (2) *Interface Specification*, which defines how *federates* (HLA compliant simulations) interact with the *RunTime Infrastructure* (RTI), and (3) *HLA Rules* governing the behavior of simulations (called federates), collections of simulations (called federations), and the RTIs.

HLA is used for engineering network-centric simulations. A layered view of HLA is shown in Figure 4.

In an HLA federation, each federate establishes a connection to the RTI. Federates do not establish direct connections between themselves; all federate-to-federate communication is through services provided by the RTI. Two interfaces define the federate-RTI connection. The RTI offers an RTI Ambassador interface to each federate. The RTI Ambassador contains all the services that are callable by federates. Likewise, a federate provides a Federate Ambassador interface to the RTI. The RTI invokes services in the Federate Ambassador when calling a federate. A federate might consist of multiple processes, but is regarded

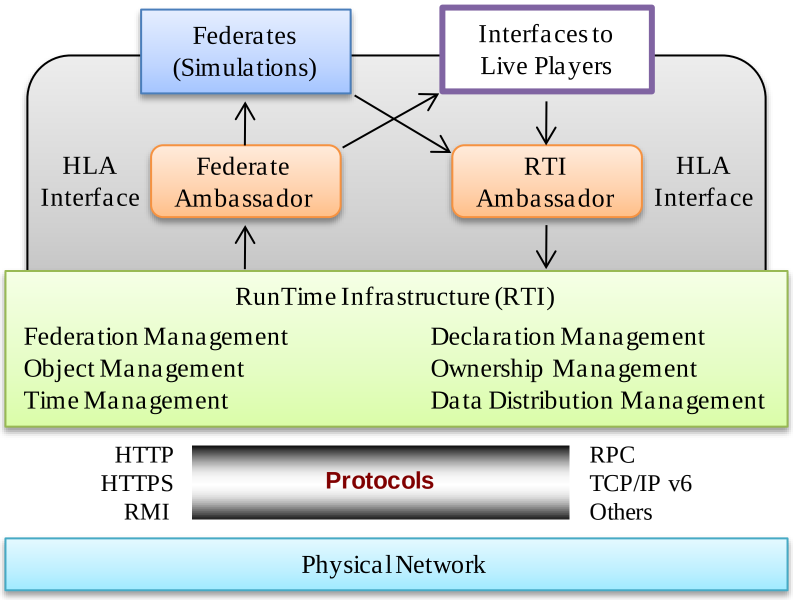

Figure 4. A Layered View of HLA

as a single federate by virtue of its single connection to the RTI. [Dahmann, Fujimoto, and Weatherly 1997]

HLA provides a broad class of services to manage the execution of federations. These services are divided into seven groups:

1. Federation management – services related to the creation and destruction of federations, the joining and resigning of federates, and certain federation-wide actions such as save and restore.
2. Declaration management – services related to publication and subscription information.
3. Object management – services related to the sending and receiving of updates and interactions.
4. Ownership management – services that allow transfer of ownership of simulation objects and attributes.
5. Time management – services that allow the explicit coordination of simulation time and event ordering.
6. Data distribution management – services that permit optimized message routing (e.g. multicasting) within the network supporting the federation.
7. Support services – miscellaneous federation support services.

Several commercial and academic RTIs have been developed including Raytheon's RTI NG Pro (www.raytheon.com), Pitch pRTI (www.pitch.se), MÄK High Performance RTI (www.mak.com), and Georgia Tech's RTI-Kit [McLean, Fujimoto, and Fitzgibbons 2004]. Although the details of their implementations vary widely, each of these RTIs provides a stateful software layer within the federation (an RTI must remember each federate's publications and subscriptions, for example).

In 2000, HLA was adopted as an IEEE Standard [IEEE 2000]. Current efforts to further the HLA standard are ongoing under the rubric of "HLA Evolved". This new standard will include support for web services interfaces.

5. CONCLUDING REMARKS

We expect more and more M&S applications to be engineered as network-centric especially for simulations used for training geographically dispersed people. Network-centric simulations can also be effective for problem solving and educational purposes by enabling access over the network. Two industry standards (Java EE and .Net Framework) and the DoD/IEEE standard (HLA) stand out as major technologies to employ for engineering network-centric simulations.

Network-centric M&S application development should also benefit from the emerging cloud computing and virtualization technologies. *Cloud computing* is a style of computing in which dynamically scalable and often virtualized resources (e.g., servers, desktops, laptops, operating systems, applications, services, storage, and telecommunications) are provided as a service over a network (e.g., Internet, virtual private network, wireless network). The following eight categories of virtualization should be considered: application server virtualization, application virtualization, hardware virtualization, management virtualization, network virtualization, operating system virtualization, service virtualization, and storage virtualization.

REFERENCES

Dahmann, J.S.; R.M. Fujimoto; R.M. Weatherly. 1997. “The Department of Defense High Level Architecture,” In *Proceedings of the 1997 Winter Simulation Conference*, ed. S. Andradóttir, K. J. Healy, D. H. Withers, and B. L. Nelson, IEEE, Piscataway, NJ, 142-149.

Fujimoto, R.M. 2000. *Parallel and Distributed Simulation Systems*, John Wiley and Sons, New York, NY.

IBM. 2009a. DB2 Product Family, http://www-01.ibm.com/software/data/db2/

IBM. 2009b. Rational Application Developer for WebSphere Software, http://www-01.ibm.com/software/awdtools/developer/application/

IEEE. 2000. IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules. IEEE Standard No. 1516-2000.

Kuhl, F.; R. Weatherly; J. Dahmann. 1999. *Creating Computer Simulation Systems – An Introduction to the High Level Architecture*, Prentice Hall, Saddle River, NJ.

McLean, T.; R.M. Fujimoto; B. Fitzgibbons. 2004. “Middleware for real-time distributed simulations,” *Concurrency and Computation: Practice & Experience 16*, 15 (Dec.), 1483 - 1501.

Microsoft. 2009a. Microsoft platform, .NET Framework, http://www.microsoft.com/net/

Microsoft. 2009b. Microsoft SQL Server, http://www.microsoft.com/sqlserver/2008/en/us/

Myers, D.S. and O. Balci. 2009. “A Web-Based Visual Simulation Architecture,” *International Journal of Modelling and Simulation 29*, 2.

Oracle. 2009. Oracle Database, http://www.oracle.com/database/

PostgreSQL. 2009. PostgreSQL Database Management System, http://www.postgresql.org

Sabah, M. and O. Balci. 2005. “Web-based Random Variate Generation for Stochastic Simulations,” *International Journal of Simulation and Process Modelling 1*, 1-2, 16-25.

SISO. 2009. *Proceedings of the Simulation Interoperability Workshops*, Simulation Interoperability Standards Organization (SISO), http://www.sisostds.org/

Sun. 2009a. Java platform, Enterprise Edition (Java EE), http://java.sun.com/javaee/

Sun. 2009b. Java EE 5 Compatible Application Servers, http://java.sun.com/javaee/overview/compatibility.jsp

Sun. 2009c. Java platform, Enterprise Edition v 5.0 Application Program Interface (API) Specifications, http://java.sun.com/javaee/5/docs/api/

AUTHOR BIOGRAPHIES

OSMAN BALCI is a Professor of Computer Science at Virginia Polytechnic Institute and State University (Virginia Tech), USA. He received B.S. and M.S. degrees from Boğaziçi University (Istanbul) in 1975 and 1977, and M.S. and Ph.D. degrees from Syracuse University (New York) in 1978 and 1981. Dr. Balci serves as ACM SIGSIM Chairman (2008-2011) and Editor-in-Chief of *ACM SIGSIM M&S Knowledge Repository*. He served as the Editor-in-Chief of two international journals: *Annals of Software Engineering* (1993-2002) and *World Wide Web* (1996-2000). He currently serves as an Area Editor of *ACM Transactions on Modeling and Computer Simulation* and Modeling and Simulation (M&S) Category Editor of *ACM Computing Reviews*. He served as an elected Director at Large of the Society for M&S International (2002-2006). Most of Dr. Balci’s work has been funded by the U.S. Navy since 1983. From 1998 to 2004, he provided technical services for the U.S. National Missile Defense and Missile Defense Agency programs in the areas of M&S verification, validation and accreditation (VV&A) and complex system independent verification and validation (IV&V). His current areas of expertise center on network-centric software engineering; architecting network-centric software-based systems; software IV&V; M&S methodology; M&S VV&A, testing, certification, credibility assessment, and quality assessment. His e-mail and web addresses are <balci@vt.edu> and <http://manta.cs.vt.edu/balci>.

ERNEST H. PAGE is a member of the technical staff for The MITRE Corporation. He received the Ph.D. in Computer Science from Virginia Tech in 1994. He serves on the editorial boards of SCS Simulation, SCS Journal of Defense Modeling and Simulation, and the Journal of Simulation. He has served as the ACM SIGSIM representative to the WSC Board of Directors since 2001. His email address is <epage@mitre.org>.