2006 CCRTS
"The State of the Art and the State of the Practice"

Progressing Toward a Net-Centric DoD: Leveraging Lessons Learned from Distributed Simulation Experiences

Topics: C2 Concepts & Organizations, Lessons Learned, C2 Experimentation

R. Douglas Flournoy
The MITRE Corporation
202 Burlington Road
Bedford, MA 01730
781-271-2774
rflourno@mitre.org

Dr. Prem Jain
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
703-983-1730
pjain@mitre.org

Elizabeth Lee
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
703-983-2692
elee@mitre.org

Robert Mikula
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
703-983-7168
rmikula@mitre.org

David Seidel
The MITRE Corporation
7525 Colshire Drive
McLean, VA 22102
703-983-6828
dseidel@mitre.org

Abstract

Programs, such as Net-Enabled Command Capability (NECC), implementing net-centric operational concepts require distributed and concurrent software development and integration/testing capabilities. Their new systems, often integrated with legacy software, have to interact with distributed external environments and users, and must execute in real time. In the commercial sector, eBay and Amazon are pioneering sandbox methods to test and debug real time performance and security-related problems. These methods need to be extended to meet DoD's unique multi-level security needs and real-time requirements.

Fortunately, a decade of DoD distributed simulation experience can potentially be applied to find an acceptable approach. Distributed and concurrent software development, testing and legacy software migration problems were overcome with a growth of techniques, processes and experiences. At the highest level, the lessons learned by the simulation community include: (1) continuously improve the systems engineering process, (2) evolve middleware standards, and (3) support the process with specialized distributed test and integration tools.

This paper provides a historical perspective for the development of the distributed simulation capability, related middleware evolution and the Federated Development and Engineering Process (FEDEP). FEDEP outlines the systems engineering steps to plan, develop, integrate and test a distributed simulation.

1. Introduction

A major challenge facing today's Department of Defense (DoD) is to provide sufficient interoperability throughout a rapidly evolving Command and Control (C2) enterprise requiring concurrent software development at disparate sites. Traditional software interface approaches that served well in the past are proving too rigid and costly to keep up with today's increasingly complex and dynamic information exchange requirements. To meet these needs, the DoD is embracing a strategy for transformation to net-centric operations.

Successful public internet services run by eBay, Amazon, Travelocity, and other companies as well as the DoD distributed simulation community have a wealth of both cultural and technical systems engineering guidance to offer for these new challenges. Commercial companies provide emulated sandboxes to distributed software developers over a network for early verification before production use of the developed software. The DoD distributed simulation community needs to address additional security and close loop real-time performance concerns in its methodology above and beyond the challenges currently faced by web service providers.

This paper reviews the successful transformation experiences of the distributed simulation community over the past decade as they relate to the challenges facing the DoD net-centric transformation, and offers some suggestions from these experiences. It identifies the need for continuously evolving a systems engineering process supporting concurrent development/ test and a need for a set of distributed test tools.

We begin by providing a brief overview of the DoD net-centric strategy and associated challenges in Section 2. Section 3 includes the history of distributed simulation and end-to-end systems engineering processes. It also includes advances in distributed test and debugging methods and tools. Section 4 includes a set of process-focused lessons learned which could guide the net-centric development of joint programs. Section 5 discusses some of the development challenges, which must be addressed for concurrent software development. Section 6 concludes this paper.

2. Transformation Programs

Today's C2 enterprise is organized around systems. By developing capabilities along self-contained system lines, the development, integration, and testing of these capabilities is simplified. However, modern warfare is evolving in a way that is exposing a weakness in the current system-based approach: the rigidity and cost inherent in developing and maintaining software interfaces between the systems.

Recognizing the need to increase the timeliness and quality of its data sharing abilities to achieve information superiority, the DoD has announced a net-centric data strategy for creating net-centric systems that use a service oriented architecture (SOA). [1] DoD has multiple efforts such as the Global Information Grid (GIG) and Net-Centric Enterprise Services (NCES) that enable this SOA paradigm and facilitate programs such as Net-

Enabled Command Capability (NECC) which bridges the community-ready solutions to fully functional net-ready DoD initiatives.

The GIG is a "globally interconnected, end-to-end set of information capabilities, associated processes, and personnel for collecting, processing, storing, disseminating and managing information on demand to warfighters, policy makers, and support personnel." [2]. Core Enterprise Services (CES) are the underlying toolset that allow SOA applications to reside and function on the GIG. Using CES, SOA applications provide APIs for one another and these services together comprise NCES. NCES aims to provide ubiquitous access to timely, secure and quality information for decision making. NCES, CES, and GIG enterprise services all refer to software in discussing SOAs.

In the NECC program, multiple communities of interest such as Time Sensitive Targeting (TST) are developing new mission threads consuming the information produced by different DoD Services. The NECC program will exploit inherent concurrency in mission threads along with the concurrency available in net-centricity to improve operational efficiencies and effectiveness such as TST timelines.

2.1 Transformation Challenges

In the current system-centric environment we have finite system boundaries and clear lines of authority; however in a net-centric environment the information space will be boundless with no single controlling authority. In a system-centric environment, requirements are known and relatively static; in net-centric operations we will have increasing dynamic requirements. Unpredictable numbers of users will be engaging in unpredictable ways at unpredictable intervals. Hence, net-centric services must be designed to be agile to respond to constantly changing conditions. [3]

Early adopters of service-oriented net-centric practices in the military domain have struggled with the issues of governance, development and acquisition, and rapid technological advances. In addition, individual services within SOAs may be chained together, yielding emergent capabilities unforeseen by the original architects. Efficient strategies for testing are needed to answer questions associated with the deployment of these architectures, including service certification, performance testing, and interoperability among communities of interest.

It appears that the service-oriented successes of commercial service providers over the public internet may only go so far to solve challenges in the military domain. And as we solve these challenges and begin to migrate some systems to service-oriented approaches, other legacy systems that are still mission-critical will need to "continue to play" with traditional or interim interface capabilities.

2.2 Net-centric Test Challenges

In a guest editorial to the ITEA Journal of Test and Evaluation titled, "Wanted: A New Test Approach for Military Net-centric Operations," David J. Carstairs highlighted the challenges facing the DoD:

The challenge of maintaining interoperability is paramount. "If we consider the generic enterprise as three increasingly complex levels, we can see how [Test and Evaluation] becomes increasingly problematic. At the lowest level, optimizing individual programs or systems is straightforward. The second level increases T&E complexity because systems are combined into a system-of-systems in which interoperability is critical. The third level, the enterprise, is the most complex and the level at which joint and coalition operations are conducted. Current T&E concepts do not scale to this level because they do not address the many possible interdependencies among the complex systems in a C^2 enterprise." [4]

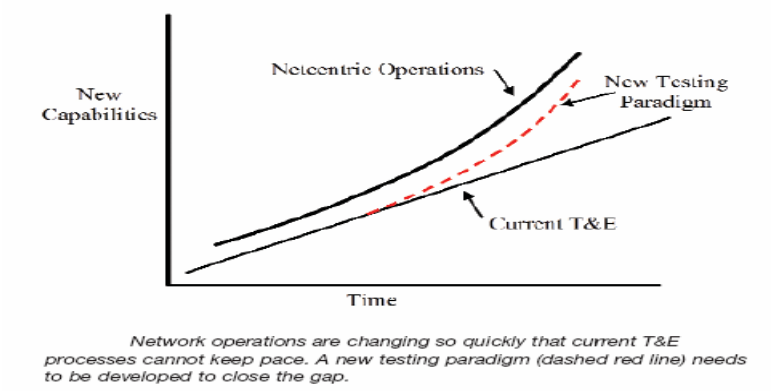

Figure 1: The Need for a New Testing Paradigm to Support Netcentric Operations

In order to support mission threads, net-centric services must be chained together forming a very complex web of interdependences that must be supported and tested. Carstairs asserts that while testing all mission threads may be impossible as expressed in Figure 1, a dedicated permanent infrastructure that supports a new paradigm for T&E may help to bridge this gap. This can be based upon the virtual environment, or could be a completely independent facility. Regardless, the main purpose remains the same, to avoid the time and expense of assembling specialized test environments for each proposed service in net-centric operational scenarios. However the question remains: what methods and tools should comprise this "new testing paradigm?"

2.3 DoD Development Initiatives

Several net-centric development and exploration environments are emerging that could gain traction implementing the DoD net-centric strategy. These pilot environments include:

The Federated Development and Certification Environment (FDCE): The Defense Information Systems Agency (DISA) is currently working on a prototype environment to manage the testing and certification of federated development for the NECC program (currently in Milestone B of the 5000.2 Defense Acquisition framework). As shown in Figure 2, it brings together developers, users, testers and certifiers of distributed software developers for the entire process from beginning to end. FDCE is organized into communities of interest that share the same needs and purposes. It facilitates interactions among community-ready and net-ready stakeholders. Community ready stakeholders define the joint operational concepts as a set of mission threads and the systems needed to achieve operational efficiencies and effectiveness. Net-ready stakeholders assure interoperability at technical and syntactical levels and are often COTS vendors. The communities each set their own criteria for certification and decide which services will be tested and certified for their needs. FDCE keeps track of this process and the players. FDCE facilitates certification. [5]

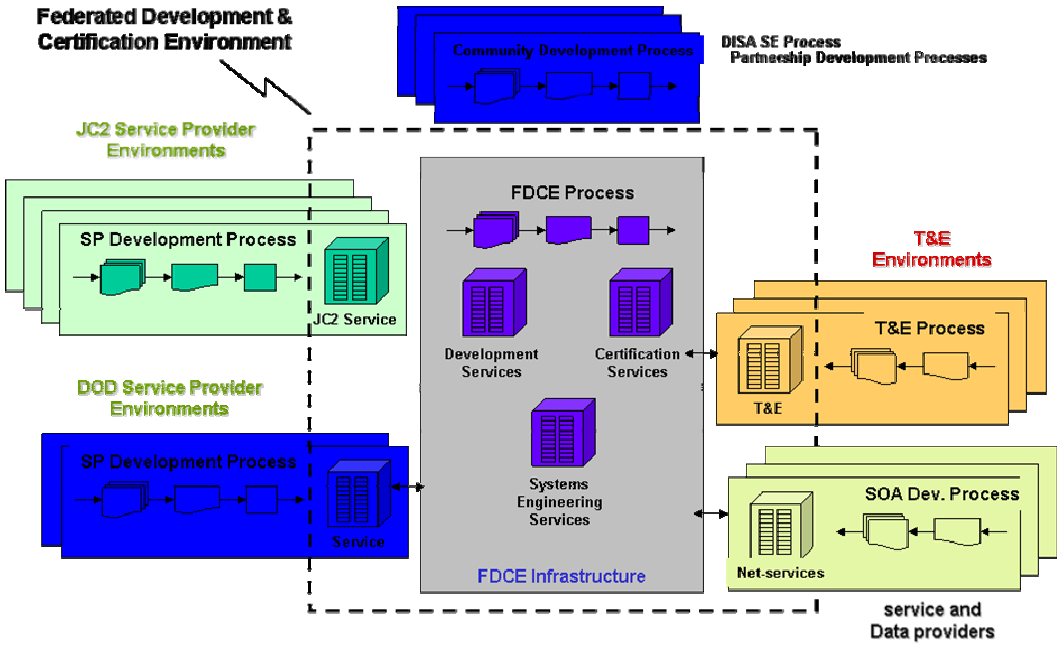

Figure 2: NECC FDCE Architecture

Net-Centric Collaboration Environment (NCCE): The NCCE is also in early stages of development and is being funded by the Air Force Electronic Systems Center (ESC) as an essential enabler for ESC delivery of net-centric capabilities. Its goal is to facilitate program development, integration, and assessment by providing a common and consistent enterprise infrastructure foundation for mission developers to access rather than developing their own infrastructure. Figure 3 is a NCCE representation of the potential integrations among Mission Services, Enterprise Infrastructure and simulations of the unavailable entities during test. [6]

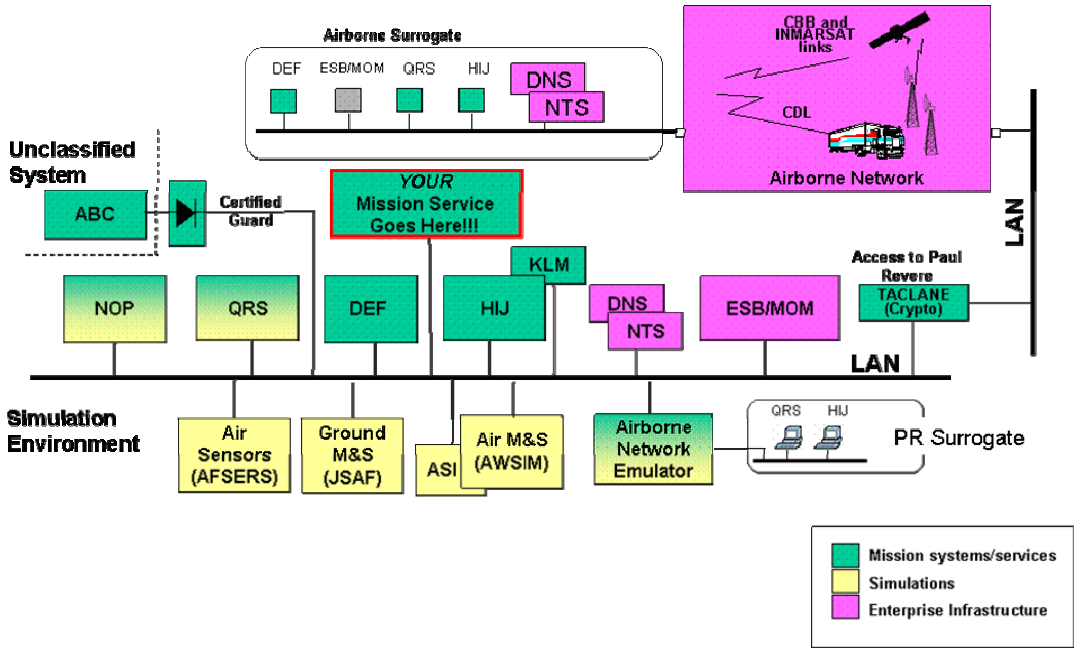

Figure 3: NCEE Architecture

Joint Information Assurance Test Suite (JIATS)—Web Enabled Test (WET): The JIATS-WET is an environment that is being developed to provide the Test & Evaluation (T&E) community with a net-centric performance, conformance, and interoperability testing capability. The JIATS and WET programs are in the early stages and are both being funded by the Centralized T&E Investment Program (CTEIP) Joint Improvement and Modernization (JIM) project. [7]

3. Development of Distributed Simulation Capability

This section provides a historical perspective of the development of distributed real time simulation capabilities and expands upon specific development of middleware, systems engineering processes with refined test methods and related tools.

3.1 Relevant History

During the mid-1980s through the mid-1990s, the military simulation community faced the need for a similar interoperability transformation requiring concurrent software development at disparate sites. At that time there was a growing need for simulation support to meet rapidly evolving requirements for training, testing, experimentation, and analysis using existing simulation software developed for specific tasks. In many cases, no single existing simulation could adequately meet the requirements of a particular event or experiment, but combinations of multiple simulations running together could meet the needs if configured to exchange certain data elements (platform locations, command messages, battle results, etc.). For example, a Joint battlestaff training exercise might require a “federation” of three simulations: an Army-accredited simulation to provide representations of ground troops and vehicle movement, a Navy-accredited simulation for

surface ships and Naval aircraft, and an Air Force simulation to “play” the rest of the air picture.

As the demand for simulation interoperability increased, it became apparent that custom, point-to-point interfaces between the simulations were costly, relatively inflexible, and offered little opportunity for reuse in future events. What was needed was a standardized mechanism for data publishing and subscribing between members of a simulation federation. And since there were extreme timing and performance requirements in many of the simulation support use cases, this data exchange framework had to make efficient use of network resources. In short, the simulation community realized that it needed to move toward a collaborative framework if they were going to support rapidly evolving user needs.

Since that time, the simulation community has succeeded in meeting its goals by adapting a common middleware in the mid-1990s. Along the way, the simulation community found that new test methods and tools were needed to address the challenges of testing in a distributed environment. After years of experience with simulation federations, a community-wide systems engineering process emerged for developing and using these federations.

3.2 Middleware Evolution

One common ingredient across successful distributed simulation projects has been the use of standardized middleware. Initially, interoperability success was achieved by using standard software-supported protocols such as Distributed Interactive Simulation (DIS). DIS offered a fully-defined message set and freely available software to broadcast these messages via UDP to other applications in a LAN environment. [8]

However, the broadcast nature of DIS makes it inefficient from a network usage perspective for many use cases. So to address these inefficiencies with publish and subscribe capabilities and additional network support options, the High Level Architecture (HLA) was introduced in the mid-1990s. HLA also introduced a more flexible framework for user definition of data fields in exchanged messages, and time synchronization services to allow simulations to run in a time-synchronized manner in non-real-time situations. [9] TENA followed quickly on the heels of the HLA thrust, offering a similar approach for runtime data exchange. [10]

The High Level Architecture is a standard for exchanging data among a federation of participating computer applications. An HLA federation is a coupled group of participants, or federates. As shown in Figure 4, these federates can be simulations, utility applications such as runtime monitors and data loggers, or interfaces to real-world systems. Federates are connected to each other by an HLA-compliant middleware package called a Runtime Infrastructure (RTI).

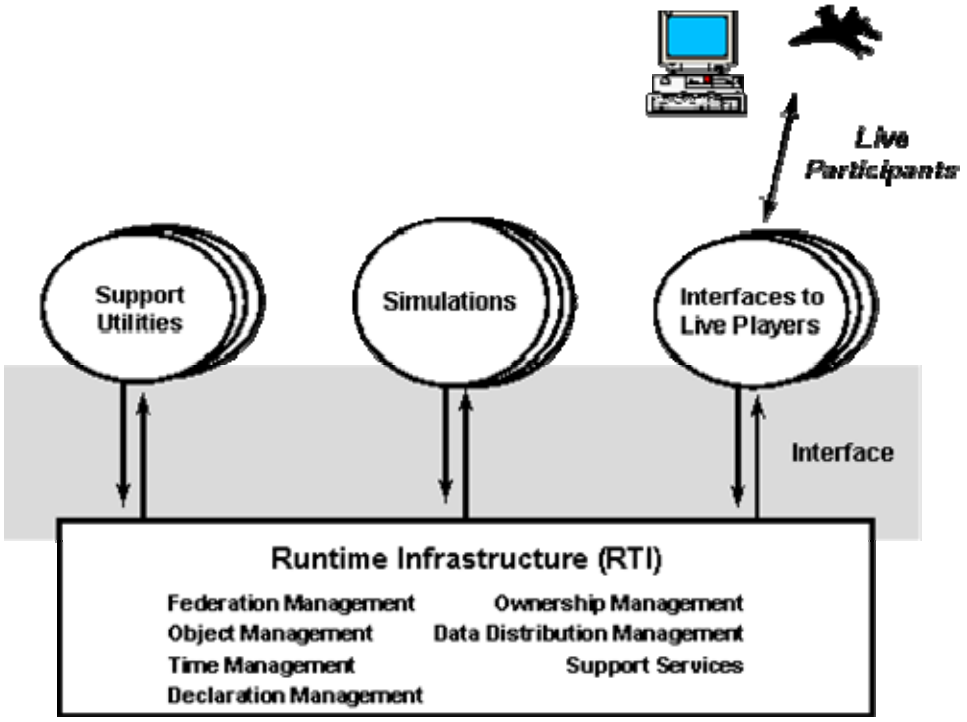

Figure 4: The High Level Architecture Federation Concept

Federates work together over a local or wide-area network to share data elements pre-defined in a Federation Object Model (FOM). To do this, they employ infrastructure services provided by the RTI in the following areas:

- Federation Management
- Declaration Management
- Object Management
- Ownership Management
- Time Management
- Data Distribution Management
- Support Services

During a typical federation execution, the RTI brokers the exchange of data between federates based on publication, subscription, and other related services. The RTI also maintains a central simulation time clock as the federation advances forward in time.

3.3 FEDEP Process

After years of collective experience using architectures like HLA and TENA, the simulation community realized that successful efforts were following a general but effective systems engineering process for employing these architectures. Eventually, HLA users developed a standardized process called the Federation Development and Execution Process (FEDEP) that captured the fundamental steps to success for

developing and deploying HLA federations. [11] The FEDEP is illustrated in Figure 5, below. These documented processes, based on years of community-wide lessons learned, served as important "stakes in the ground" marking the successful transformation of the simulation community to mature service-based approaches. Now, simulation engineers can use these processes to guide their federation efforts rather than relearning lessons the hard way.

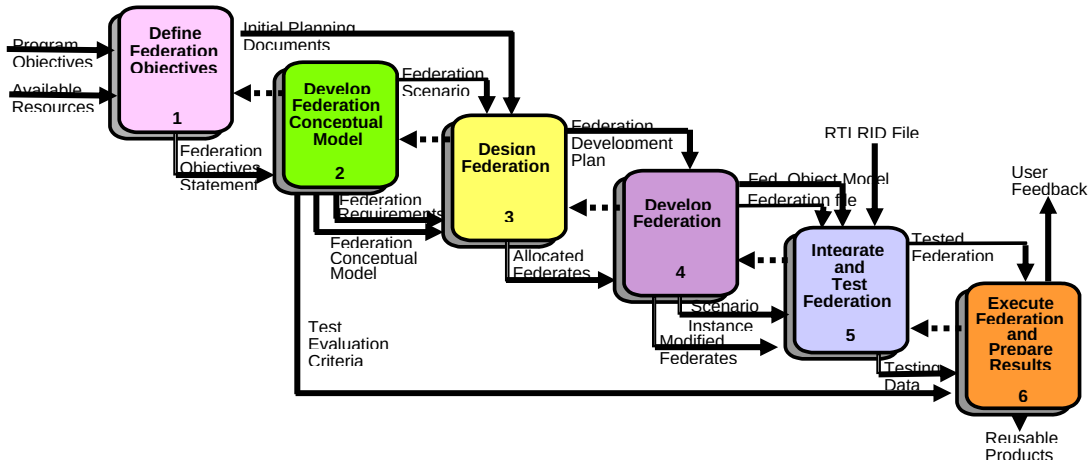

Figure 5: Federated Development and Execution Process

The six steps below, taken directly from [16], outline the FEDEP process.

Step 1: Define Federation Objectives. The federation user and federation development team define and agree on a set of objectives and document what must be accomplished to achieve those objectives.

Step 2: Develop Federation Conceptual Model. Based on the characteristics of the problem space, an appropriate representation of the real world domain is developed.

Step 3: Design Federation. Federation participants (federates) are determined, and required functionalities are allocated to the federates.

Step 4: Develop Federation. The Federation Object Model (FOM) is developed, federate agreements on consistent databases/algorithms are established, and modifications to federates are implemented (as required).

Step 5: Integrate and Test Federation. All necessary federation implementation activities are performed, and testing is conducted to ensure that interoperability requirements are being met.

Step 6: Execute Federation and Prepare Results. The federation is executed, outputs are generated, and results are provided.

In step one, the common purpose of the federation is determined. In step two, the conceptual model is created, which gives the federation a concrete plan by which to meet the objectives. In step three, each participant and their roles are determined. In step four, the FOM, which contain the specifics of data exchange and formats, is created. The

federation may have to come back to this step numerous times before settling on the final FOM. Federations continue to evolve cycle through steps 4, 5 and 6 for the duration of the federation's existence.

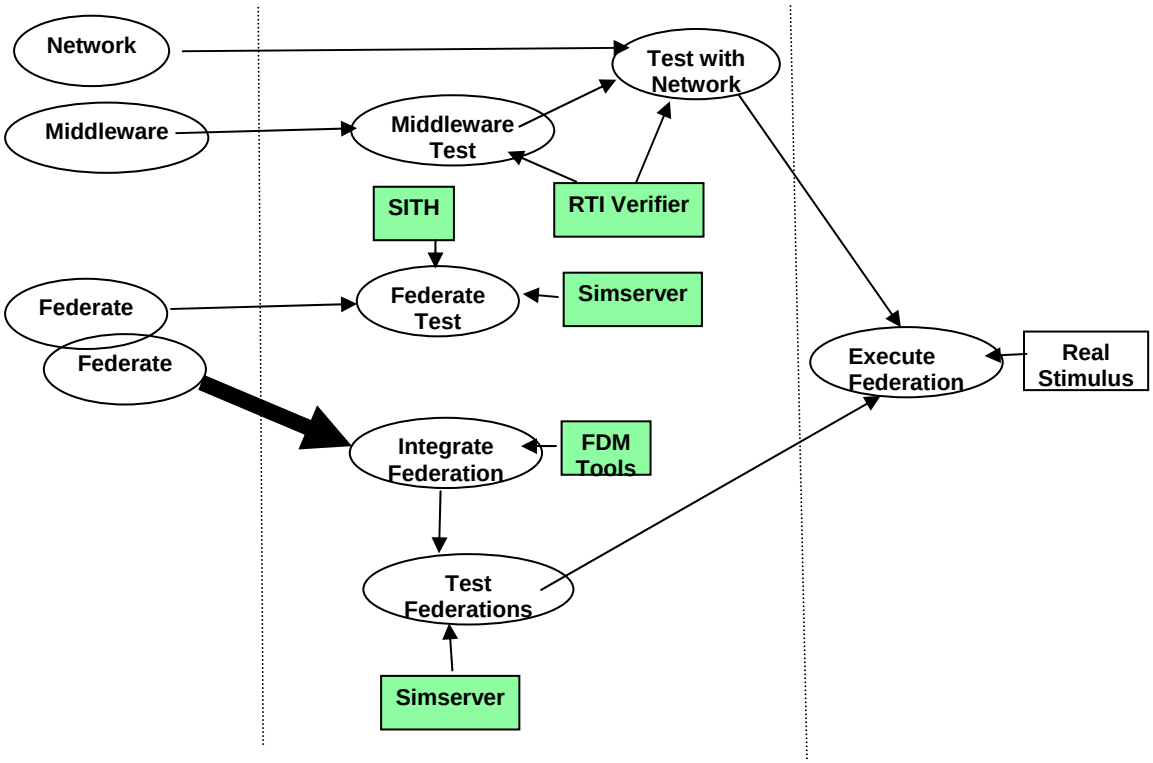

Figure 6: Details of the Integration and Test step of FEDEP

3.4 Integration and Test FEDEP Process: Figure 6 show details of step 5 in the FEDEP. The process allows concurrent development of hardware and software such as network infrastructure, middleware and a collection of application software within a federate. As shown in the Figure 6, testing is done at multiple levels and tools are needed to create test stimuli. Test and integration tools, annotated in the process diagram in green, are briefly described in the next section.

3.5 Integration and Test Tools

Just as the DoD-wide net-centric transformation is introducing new challenges for testing data exchanges within the C2 enterprise, the services introduced by HLA and TENA forced new test approaches to the forefront in the simulation community. Traditional systems-based testing that isolated capabilities within simulations gave way to new methods designed to test multiple combinations in threads or "chains" of networked execution.

As with net-centric systems of systems, federation testing can also be costly and logistically complex. To help alleviate some of this complexity, MITRE developed a set of systems engineering and testing approaches and tools for federations of simulations:

RTI Verifier: The RTI Verifier is a tool developed to meet the needs of the Defense Modeling and Simulation Office (DMSO) to evaluate conformance of RTIs to the HLA specification. The RTI Verifier uses a requirements-based approach to testing. The most interesting aspect of the RTI Verifier architecture is that the testing software does not stimulate the RTI directly. Based on simple scripts created by a test engineer, the RTI Verifier spawns a number of virtual testing federates which interact with the RTI during testing to determine if the RTI behaves correctly. The RTI Verifier consists of a database housing all of the tests and test results; a Launcher that starts five federates and the RTI in test; and a Test Controller that drives the federates to interact with the RTI. Other supporting components include Script Definition Language (SDL) to specify tests. [12]

Simulation Interoperability Test Harness (SITH): The SITH was developed with the goal of testing not just the RTI but the individual federates and overall federation performance. The SITH also employs the approach of spawning virtual federates that connect to the RTI. These federates perform as directed in easily created scripts to publish, subscribe and otherwise serve as counterpart federates. Using these stand-ins federates the behavior and data exchanges of the federate under test can be examined in a federation environment. This concept of a virtual stand-in federate whose behavior and messages are translucent through the GUI of the testing software (RTI Verifier or SITH) has proven very successful in testing many federations including the Joint Simulation System (JSIMS) and the Army Constructive Training Federation (ACTF). [13]

SimServer: Simulated data streams have long been employed to support prototyping and experimentation. These data streams create the operational context within which systems and concepts are demonstrated, tested, integrated, and exercised. By employing a select set of web-inspired computing techniques, SimServer is providing on-demand access to simulated data streams. At the SimServer web site, consumers plan, configure, execute, and monitor their data streams. Rather than developing capabilities from scratch, projects use the site to browse available simulation services and reuse or modify them. This common repository of tailorable, on-demand simulation services frees more project effort to be devoted to prototyping and experimentation activities, facilitating broader and deeper experimentation programs that deliver richer insights for shaping the future of fielded systems. [14]

Federation Data Management: MITRE has also developed a family of federation data management (FDM) tools to support the federate and federation integration activities. These tools began with the development of Basic Interface Tests (BITs) which identify the Federation Object Model (FOM) objects a federate publishes and the interfaces surrounding and related to those objects. The FDM tools verified the existence of the object in the FOM as well as identified objects not yet in the FOM, thus determining needed revisions. In addition, the tools identified subscribing and publishing federates as well as possible mismatches.

The FDM tools were also used to populate the BITs with the publication and subscription actions expected of the other federation members. As the FDM evolved a Basic Interface Test (BIT) Tracking Tool (BTT) was developed. The BTT records integration status for each BIT step and provides reports by BIT, by federate, or by federation enclave. The BTT provides tabular or graphical comparison of planned vs. actual integration progress. Data within the BTT can be analyzed to assess the status of interfaces to C4I systems, or to assess progress in meeting operational requirements. [15]

3.5 Summary

Distributed simulation capability broke new ground and evolved as the need arose. The evolution steps were the evolutionary middleware standards, FEDEP systems engineering process and the refinement of integration and test steps. Throughout the evolution, a set of internally developed tools and COTS provided the needed automation.

4. Lessons Learned

In hindsight, DoD distributed simulation could have been realized more quickly through 1) early availability of well-supported FEDEP processes built around multiple levels of testing and 2) early recognition of the need for evolving middleware standards and specialized distributed integrated test tools. These learned lessons are described below.

4.1 Continuously improve systems engineering process

Based on the impact of the FEDEP as a guiding systems engineering standard, we recommend that the net-centric community begin developing such an engineering process. As the community determines "what works" and "what doesn't" through experience, this process can be refined to become an invaluable net-centric capability development roadmap. The history of developing distributed simulations using the FEDEP offers these and many more useful guideposts directly applicable to the development of net-centric federations of services:

- Support multiple levels of tests
- Emphasize the use of development, integration, test and debug tools
- Use an overarching systems engineer to oversee and broker development, testing, and integration. [13]

4.2 Plan for evolving middleware standards

While the use of middleware standards was perhaps the first and most important step toward the current successful state of simulation interconnectivity, convergence on a single middleware standard across the entire simulation community has not been reached. However, in hindsight, efforts to mandate a single data exchange approach were not necessary—the community now accepts several different middleware solutions along with bridging between these middleware solutions when multiple middleware schemes are necessary in the same federation.

Based on simulation community middleware successes, we recommend the GIG community embrace and explore Enterprise Service Buses (ESBs) and drive the marketplace to some level of standardization in this area. However we do not recommend that the community attempt to converge on a single solution—instead, they should make sure that multiple candidate solutions can interconnect effectively within the enterprise.

4.3 Develop specialized distributed test and integration tools

The simulation community conquered testing challenges by developing test management tools that can be scripted to emulate individual services or federations of services. Analogous tools such as SOATest are emerging in the net-centric community; we recommend such tools be embraced and evolved to provide much-needed comprehensive net-centric test capabilities.

Web-based simulation tools are becoming available that may directly support the DoD net-centric transformation by helping meet expanding needs for testing, analysis, and experimentation support. The Network Centric Operations Industry Consortium (NCOIC) has recognized SimServer and the Modeling and Simulation GIG COI as potential enablers in their strategy to lower the risks and costs associated with developing and integrating net-centric systems.

5. Open challenges for distributed development

Based on past experience we discuss some of the open challenges to managing the rising complexity of concurrent disparate software development practices:

Multiple levels of interoperability: SOA provides interoperability at the middleware dealing with syntactical level. Distributed development also needs to deal with interoperability at the semantic, pragmatic and dynamic levels. These levels are specific to each project and need to be dealt by DoD programs rather than COTS vendors.

Multi-level Testing: Test complexity can be managed by recognizing multiple levels of test and getting an agreement within the industry for these levels. Acceptance of such standard levels will help the certification process, and will also help in the isolation of problems identified during system level testing.

Mission level modeling: System level testing provides a limited confidence that the developed systems can cover the wide variations implicit in the net-centric concepts. These include scalability, timing behavior due to non deterministic inputs such as queuing, and conditional paths taken in mission threads. Mission level modeling can complement system level test to deal with these shortcomings.

Efficient and consistent test stimulus developments: Testing will continue to take a larger share of the engineering resources as SOA promotes the reuse of existing software.

This creates burden on creating input stimulus for testing. The stimulus development and its maintenance are large software development efforts. The quality of this software determines the quality of validation.

6. Conclusions

Net-centric systems require concurrent software development at disparate sites, associated integration techniques, and distributed test and debugging strategies in a networked environment. These new challenges can take advantage of the experiences in distributed real-time simulation and federated development. The FEDEP systems engineering process, supported with a set of tools, has provided high engineering efficiencies within the distributed simulation community. MITRE has refined the integration and test step of FEDEP and built a number of associated tools to further support distributed testing and promote engineering efficiencies.

Programs such as NECC must evaluate these successes to guide the development of its engineering and management processes. These programs must recognize a need for concurrent development at disparate sites and implement processes supported with automation for distributed test and debugging. They must support evolution of middleware standards. They must deal with rising complexity by testing at different levels, introducing mission level modeling capability, and developing efficient and effective test stimuli to facilitate debugging methods tailored to distributed environments.

REFERENCES

[1] “Department of Defense Net-Centric Data Strategy”, May 2003

[2] Miller, A., Jefferson, M., Rogers, J., “Global Information Grid Architecture”, The Edge, MITRE Corporation. July, 2001.

[3] Foster, F., “Net-Centric Enterprise Services (NCES) Overview”, MITRE Corporation, July 2004.

[4] Carstairs, D., “Wanted: A New Test Approach for Military Net Centric Operations”, ITEA Journal of Test and Evaluation, September-October 2005.

[5] Jain, P., “FDCE Architecture & Mission Level Modeling”, MITRE Corporation, January 2006.

[6] Sargent, C., Scarano, J., “Net-Centric Collaborative Environment (NCCE), Cross Wing Outreach”, MITRE Corporation, June 2005.

[7] Napier, J., “JIATS-WET Fact Sheet”, MITRE Corporation, December 2005.

[8] IEEE Standard for Distributed Interactive Simulation—Application Protocols, IEEE Standard 1278.1-1995, 21 September 1995.

[9] IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA)—Framework and Rules. IEEE Standard 1516-2000, 21 September 2000.

[10] Powell, E. "Test and Training Enabling Architecture (TENA) Overview Course (TOC)," October 2005.

[11] IEEE Recommended Practice for High Level Architecture (HLA) Federation Development and Execution Process (FEDEP), Standard 1516.3-2003.

[12] Symington, S., Kaplan, J., Kuhl, F, Tufarolo, J., Weatherly, R., and Nielsen J.: "Verifying HLA RTIs", Simulation Interoperability Workshop, Fall 2000.

[13] Ring, K. "SITH: A General-Purpose HLA and Integration Testing Tool," Simulation Interoperability Workshop, Fall 2003.

[14] D. Flournoy, R. Mikula, D. Seidel, R. Weatherly. *SimServer: Simulated Data Streams on Demand Via the Web,* Spring '05 Simulation Interoperability Workshop (SIW) paper and presentation 05S-SIW-138, April 2005.

[15] Feinerman, L., Mikula, R., "Federation Data Management Tools to Ensure FOM, Federate, and Federation Consistency", Fall 2003 Simulation Interoperability Workshop.

[16] High Level Architecture Federation Development and Execution Process (FEDEP) Model v. 1.5, Defense Modeling and Simulation Office, December 1999.

AUTHOR BIOGRAPHIES

DOUGLAS FLOURNOY is a Principal Simulation and Modeling Engineer within the Center for Acquisition and Systems Analysis at the MITRE Corporation in Bedford, Massachusetts. Doug is exploring Web-based simulation services, engineering a Link-16 communications simulation and investigating HLA federation performance prediction methods. He also has experience in human behavior and process modeling, simulation-to-C4I system interfacing, and user interface prototyping. Doug holds a Bachelor of Science Degree in Mechanical Engineering from the Pennsylvania State University and a Master of Science Degree in Operations Research from the George Washington University.

DR. PREM JAIN is lead Simulation and Modeling Engineer within the Center for Acquisition and Systems Analysis at the MITRE Corporation in Mclean Virginia. He has spent over 20 years in Electronic Design Automation industry and has been a faculty member at University of Texas in Austin, where he guided Graduate level research for electronic architecture analysis and synthesis. Before joining BAE Systems, he founded and was the CEO of the company "Cynergy System Design", which served the "System on a Chip" architecture analysis and architectural synthesis tools. His R&D experience

includes GE and Schlumberger. He holds a Bachelor of Science in Electrical Engineering from Indian Institute of Technology Kanpur, India and Masters and PhD in EE from Rensselaer Polytechnic Institute , Troy NY.

ELIZABETH LEE is a Simulation Modeling Engineer within the Center for Acquisition and Systems Analysis of the MITRE Corporation in McLean, VA. In addition to investigating net-centric systems testing, she has previously worked on trade studies for the Future Combat Systems (FCS). Elizabeth holds a Bachelor of Science Degree in Operations Research and Industrial Engineering from Cornell University and a Master of Engineering degree in Systems Engineering from University of Maryland.

ROBERT MIKULA is a Lead Software Systems Engineer within the Center for Innovative Computing and Informatics of the MITRE Corporation in McLean, VA. He currently supports the Center for Enterprise Modernization with the implementation of Service Oriented Architectures within the federal government. Rob has experience in web-based software development, XML, human computer interfaces, databases, and object request broker technologies. He holds a Project Management Professional Certification and is currently pursuing doctoral studies at George Mason University.

DAVID SEIDEL is a Senior Principal Simulation and Modeling Engineer for the MITRE Corporation. He has participated in the development of several distributed simulation projects including Aggregate Level Simulation Protocol, HLA prototypes and standards, DMSO outreach federations, the Joint Simulation System (JSIMS) and the Joint Distributed Engineering Plant (JDEP). Mr. Seidel has a B.S. from Illinois Institute of Technology and an M.B.A. from Bryant College.