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

R. Douglas Flournoy
Dr. Prem Jain
Elizabeth Lee
Robert Mikula
David Seidel

The MITRE Corporation

CCRTS
June 21, 2006

Sponsored by Joint Interoperability Test Command (JITC)
Agenda

- Net centric challenges
- Distributed simulation capability
- Tools summary
- Lessons learned
Transformation Challenges

• Paradigm shift to data-centric from systems-centric
  – Global Information Grid (GIG)
  – Net Centric Enterprise Services (NCES)
  – Net Enabled Command Capability (NECC)

• Must
  – Unite Communities Of Interest (COI) for information sharing
  – Provide leadership to acquire net-ready COTS solutions
  – Fill the gap among community-ready and net-ready initiatives
  – Provide an evolutionary path to migrate legacy software to DoD Services

• Transition to delayed-composition of DoD Services
  – Requirements in a systems-centric acquisition paradigm are defined early and are static
  – Service Oriented Architecture (SOA) paradigm provides the potential of delayed (even at run-time) service composition
**Net Centric Test Challenges**

- **DoD rising test challenges**
  - Optimizing individual programs
  - Interoperability among system-of-systems
  - Support mission threads for joint and coalition operation

- **Must avoid the time and expense of assembling specialized test environment for each proposed DoD Service to test net centric operational scenarios**

- **Challenge: What is the new testing paradigm?**

The **Federated Development and Certification Environment (FDCE)** is the set of processes and supporting infrastructure that is needed to support the conduct of net-centric Capability Provisioning Operations.
Distributed Simulation Capability
CONOPS

Distributed Simulation History

Technologies DIS ALSP HLA TENA FEDEP

Standards

Distributed Simulators
Trainees in tank and aircraft simulators maneuver and engage each other and simulated vehicles

Test & Evaluation
Testers expand the scope of tests using simulations of system under test and test environment

CONOPS Development
Warfighters determine how to integrate new capabilities into existing forces

Performance Analysis
Analysts assess system performance for systems where live tests are too costly or dangerous

Command & Staff training
Geographically dispersed staffs employ simulated forces against a simulated enemy

Experimentation
Analysts evaluate the impact of proposed new equipment on war fighting success

Planning
Warfighters test courses of action against human or simulated opponents

Test Rehearsal
Testers simulate live test to verify planning

Training T&E CONOPS Planning Performance
Federated Development and Engineering Process (FEDEP)

1. Define Federation Objectives
2. Develop Federation Conceptual Model
3. Design Federation
4. Develop Federation
5. Integrate and Test Federation
6. Execute Federation and Prepare Results

Program Objectives
Available Resources

Initial Planning Documents
Federation Objectives Statement
Federation Requirements
Federation Conceptual Model
Allocated Federates
Scenario
Test Evaluation Criteria
Fed. Object Model
Scenario Instance
Modified Federates
RTI RID File

User Feedback
Reusable Product

Test Evaluation Criteria
Testing Data

Execute Federation and Prepare Results
Middle Ware Standard: Run Time Infrastructure

- Support Utilities
- Simulations
- Interfaces to Live Players

Runtime Infrastructure (RTI)
- Federation Management
- Ownership Management
- Object Management
- Data Distribution Management
- Time Management
- Support Services
- Declaration Management

Live Participants
Integration and Test Sub-process

Supports concurrent and disparate software development, integration and test
• Middleware verification
• Simulation data server
• Federation data management
• **RTI Verifier**
  - Evaluates conformance of RTIs to the HLA specification
  - Consists of
    - Database housing all of the tests and test results
    - Launcher starts five virtual federates and the RTI in test
    - Test controller driving the federates to interact with the RTI
    - Script Definition Language (SDL) to specify tests

• **Simulation Interoperability Test Harness (SITH)**
  - Test individual federates and overall federation performance
  - Spawns virtual federates that connect to the RTI
  - Very successful in testing many federations including
    - Joint Simulation System (JSIMS)
    - Army Constructive Training Federation (ACTF)
• Data streams create the operational context within which systems and concepts are demonstrated, tested, integrated, and exercised
• Employs a select set of web-inspired computing techniques
• Provides on-demand access to simulated data streams
• Consumers plan, configure, execute, and monitor their data streams at SimServer web site
• Rather than developing capabilities from scratch, projects use the site to browse available simulation services and reuse or modify them.
Federation Data Management

• Support the federate and federation integration activities

• Basic Interface Tests (BITs): identify
  – Federation Object Model (FOM) objects a federate publishes
  – Interfaces surrounding and related to those objects

• The FDM tools: verifies
  – Verifies existence of the object in the FOM as well as identified objects not yet in the FOM
  – Determines needed revisions and possible mismatches

• Basic Interface Test (BIT) Tracking Tool (BTT)
  – Records integration status for each BIT step and provides reports by
    • BIT
    • Federate
    • Federation enclave
  – 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
    • Assess progress in meeting operational requirements
Lessons Learned

1. Process
2. Standard
3. Automation
1. Continuously Improve the Systems Engineering Process

- Net-centric community must have a process such as FEDEP systems engineering process
  - Must allow refinement through experience
  - Must support multiple levels of tests
  - Must emphasize the use of development, integration, test and debug tools

- Use an overarching systems engineer to oversee and broker development, testing, and integration
  - Tradeoff among future interoperability and suboptimal information exchanges
  - Role played by profit or non-profit company
2. Evolve Middleware Standards

• Embrace and explore Enterprise Service Buses (ESBs)
  – Middleware standards were pivotal for current successful state of simulation interconnectivity
  – Drive the marketplace to some level of standardization
  – Do not attempt to converge to a single solution
  – Make sure that multiple candidate solutions can interconnect effectively within the enterprise
3. Plan for Distributed Test and Integration Tools

- Embrace and evolve much-needed comprehensive net-centric test capabilities
  - Script to emulate individual services or federations of services
  - Use emerging commercial SOA testing tools such as SOAPscope and SOA test
  - Use Web-based simulation tools to meet expanding needs for testing, analysis, and experimentation

- Lower the risks and costs associated with developing and integrating net-centric systems
  - Use tools such as SimServer
  - Promote the use of Modeling and Simulation
Managing Net Centric Test Challenges

• Manage each level of interoperability differently
  – SOA middleware capability provides interoperability at the syntactic level
  – DoD programs must develop standards for higher levels of interoperability via data and process strategies

• Manage test complexity by testing at multiple levels
  – Develop standards (mission, data, services) for test levels and support via automation
  – Improve certification efficiency by reusing test at different level
  – Develop plans to debug in a distributed environment (e.g. isolation)

• Model at mission level
  – Validate operational concepts early
  – Provide context for system test and evaluation
  – Improve test confidence for
    • Coverage of wide temporal variations in the net-centric concepts
    • Scalability

• Improve efficiency to develop consistent test stimulus
  – SOA paradigm requires much higher test engineering efforts due to short delivery time reusing developed services
  – Test stimulus quality determines the system validation quality
  – Treat SOA stimulus development/ maintenance as large software development project
Conclusion

• Irrespective of implementation technology evolutions there is an increasing need for a capability to develop
  – concurrent
  – disparate
  – collaborative software

• Distributed simulation community has
  – Struggled with similar challenges over a decade
  – Developed systems engineering process
  – Developed middleware standards
  – Specialized distributed integration and test tools

• In hind sight, distributed simulation adoption rate could have been accelerated by
  – Having a systems engineering process early
  – Having a plan for distributed integration and test