CS 582
Modeling and Simulation II

Distributed Simulation
Concepts and Protocols

Mikel D. Petty, Ph.D.
University of Alabama in Huntsville
Center for Modeling, Simulation, and Analysis

THE UNIVERSITY OF ALABAMA IN HUNTSVILLE

Center for Modeling, Simulation, and Analysis
Instructor biography

- Mikel D. Petty, Ph.D.
- Current position
  - Director; Center for Modeling, Simulation, and Analysis
  - Associate Professor, Computer Science
- Education
  - Ph.D. Computer Science, UCF 1997
  - M.S. Computer Science, UCF 1988
  - B.S. Computer Science, CSUS 1980
- Research
  - Modeling and simulation
  - > 175 research papers published
  - ~ $15 million total research funding awarded
Instructor Distributed Simulation experience

• Selected Dist Sim research projects
  ▪ 1992: Multi-resolution modeling
  ▪ 1992: I/ITSEC DIS interoperability demo
  ▪ 1996: HLA Platform Proto-Federation
  ▪ 1997: DIS–HLA protocol translator
  ▪ 1997: HLA Data Distribution Management services
  ▪ 1998: HLA medical federation
  ▪ 2000: IEEE 1516-2000 HLA standard
  ▪ 2003: Crowd model in HLA
  ▪ 2010: IEEE 1516-2010 HLA standard
  ▪ 2013: IEEE 1730-2013 DMAO standard

• ~90 Dist Sim publications 1995-2012
Overall conceptual outline

- **Concepts and Protocols**
  - What is Dist Sim?
  - Why is Dist Sim useful?
  - How does Dist Sim work, in concept?
  - What Dist Sim protocols have been developed?

- **HLA Development**
  - How can a Dist Sim be implemented using HLA?
  - What tools and products support HLA development?

- **Applications and Case Studies**
  - What applications are appropriate for Dist Sim?
  - How has Dist Sim been used successfully?
Primary sources


Many other secondary sources (books, papers, and reports).
Concepts and Protocols outline

- Motivation and introduction
- Background definitions and concepts
- Semi-automated forces
- Distributed simulation definitions and concepts
- Distributed simulation protocol standards
- Close Combat Tactical Trainer
- Introduction to interoperability protocols
- Distributed Interactive Simulation
- High Level Architecture

(continued on next slide)
(continued from previous slide)

- Test and Training Enabling Architecture
- Implementing interoperability
- Terrain issues in distributed simulation
- Summary and references
Motivation and introduction
Motivation and objectives

- **Motivation**
  - Distributed simulation (Dist Sim) widely used in DoD
  - Many important models use Dist Sim
  - Dist Sim protocols embody key M&S ideas

- **Objectives**
  - Understanding of Dist Sim definitions and concepts
  - Knowledge of important Dist Sim protocols
  - Exposure to key Dist Sim models and systems
M&S architectures

- Architecture: standalone vs distributed
  - Standalone
    - Single program running on single computer
    - More common
    - Less complexity and implementation effort
    - Less validation effort
  - Distributed (aka networked, interoperable)
    - Multiple programs running on multiple computers, linked via network and protocol
    - Less common
    - More complexity and implementation effort
    - More validation effort
Example distributed simulation: America’s Army

- Recruiting and familiarization tool for U. S. Army
- Multiplayer online game, linked via Internet
- First person shooter
- 9.7M registered users, 42.6M downloads (2010)
Advantages (benefits) of distributed simulation

- Access additional computational power
  - Multiple computers
- Support multiple users or participants
- Combine heterogeneous models
  - Developed for different purposes
  - Implemented by different developers
- Exploit existing models and model federations
  - Interoperability protocol allows connection and use
- Support geographic separation of users
- Allow multiple security levels in single simulation
  - Models run at different security levels
Background definitions and concepts
Concepts

- Model: representation of something else
- Simulation: executing a model over time

\[ R = 2.59 \times 4 \sigma \times \sqrt[4]{\log^{-1}\left(\frac{ERP_r}{10}\right) \log^{-1}\left(\frac{G_r}{10}\right) \log^{-1}\left(\frac{MDS_r}{10}\right) \log^{-1}\left(\frac{FEL_r}{10}\right) F_i^2} \]
Definition

**Model**: A physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process.  
[DOD, 1996] [DOD, 2009]

- Representation of something else, often a “real-world” system
- Some aspects of the modeled system are represented in the model, others not
Example model

Equation describing vertical height of an object moving under gravity.

\[ h(t) = -16t^2 + vt + s \]

- \( h \) = height (feet)
- \( t \) = time in motion (seconds)
- \( v \) = initial velocity (feet per second, + is up)
- \( s \) = initial height (feet)

Note that at \( t = 0 \), \( h = s \), as expected.
$h(t) = -16t^2 + vt + s$

Model does represent
- Height of object (output of model)
- Mass of earth (as the $-16$ coefficient)
- Initial state, as velocity $v$ and height $s$

Model does not represent
- Air resistance (not included in model)
- Location (assumed to be near surface of earth)
- Mass of object (not included in model)
Definition

**Simulation:** Executing a model over time. Also, a technique for testing, analysis, or training in which real world systems are used, or where a model reproduces real world and conceptual systems. [DOD, 1996] [DOD, 2009]

Alternative uses of term (to be avoided)
- A large composite model
- Software implementation of a model
Example simulation

Model: \( h(t) = -16t^2 + vt + s \)

Data: \( v = 100, s = 1000 \)

<table>
<thead>
<tr>
<th>( t )</th>
<th>( h(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1000</td>
</tr>
<tr>
<td>1</td>
<td>1084</td>
</tr>
<tr>
<td>2</td>
<td>1136</td>
</tr>
<tr>
<td>3</td>
<td>1156</td>
</tr>
<tr>
<td>4</td>
<td>1144</td>
</tr>
<tr>
<td>5</td>
<td>1100</td>
</tr>
<tr>
<td>6</td>
<td>1024</td>
</tr>
<tr>
<td>7</td>
<td>916</td>
</tr>
<tr>
<td>8</td>
<td>776</td>
</tr>
<tr>
<td>9</td>
<td>604</td>
</tr>
<tr>
<td>10</td>
<td>400</td>
</tr>
<tr>
<td>11</td>
<td>164</td>
</tr>
</tbody>
</table>
Simulation vs reality

Real-world system in start state

Model in start state

Initialization

h(t) = −16t^2 + vt + s
1000 = −16(0)^2 + 100(0) + 1000

Time

Physics

Real-world system in end state

Model in end state

Simulation

Computation

Interpretation

Validation

h(t) = −16t^2 + vt + s
0 = −16(11.63)^2 + 100(11.63) + 1000

© 2013 University of Alabama in Huntsville; © 2013 Mikel D. Petty, Ph.D.
Background definitions, 1 of 2

Simuland
- Real-world system
- Thing to be simulated

Requirements
- Intended uses
- Needed validity, resolution, scale

Conceptual model [BanksC, 2010]
- Simuland components, structure
- Aspects of simuland to model
- Implementation specifications
- Use cases
- Assumptions
- Initial model parameter values

\[ h(t) = -16t^2 + vt + s \]
Background definitions, 2 of 2

Executable model
- Computer software
- Implemented conceptual model

Results
- Output of model
- Produced during simulation

```c
/* Height of an object moving in gravity. */
/* Initial height v and velocity s constants. */
main()
{
    float h, v = 100.0, s = 1000.0;
    int t;
    for (t = 0, h = s; h >= 0.0; t++)
    {
        h = (-16.0 * t * t) + (v * t) + s;
        printf("Height at time %d = %f\n", t, h);
    }
}
```
Semi-Automated Forces
Semi-automated forces (SAF) systems [Petty, 1995]

- Generate and control multiple simulated entities
- Used standalone or with other models
- Autonomous behavior for SAF entities [Petty, 2009]
  - Generated by software in SAF model
  - Controlled by human operator via user interface
  - Military hierarchy represented
- Environment represented (e.g., terrain)
- Example SAF systems
  - ModSAF; once most widely used, now unsupported
  - OneSAF; current U. S. Army standard SAF
  - VT MAK VR-Forces; commercial product
- aka computer generated forces (CGF)
Example SAF: ModSAF

- **Terrain map**
- **Entities**
- **Control measures**
- **Map commands**
- **Unit icons**
- **Execution matrix**
Example SAF: OneSAF

- Terrain map
- Map commands
- Entity hierarchy
- Execution matrix
- Entity status
Example SAF: VR-Forces
Example SAF: VR-Forces

VR-Forces Operator Interface
Recreation of 1991 Battle of 73 Easting
[Daniels, 2012]
Distributed simulation definitions and concepts
Definition

**Distributed simulation.** Multiple collaborating simulations distributed across locations, computers, and/or processes.

Distributed simulations typically
- Cooperatively simulate simuland
- Each simulates some portion of simuland
- Exchange data about simuland via network messages
Generic distributed simulation architecture

- $F = \{f_1, f_2, \ldots, f_m\}$ simulation nodes
- $O = \{o_1, o_2, \ldots, o_n\}$ simulated entities
- $R$ non-simulation support node
- Network and protocol
Distributed simulation conceptual framework

- Simulation domain
  - What entities are represented?
- Model characteristics
  - How are the entities modeled?
- Software architecture and implementation
  - What implementation techniques are used?
- Simulation infrastructure
  - What network services & protocol are used?
- Computing infrastructure
  - What computing hardware is used?
Distributed simulation system components

- Models/Simulations (simulation nodes)
- Utilities (non-simulation support nodes)
- Network and protocol
Models/Simulations

Node of a distributed simulation system that is simulating part of the exercise/experiment

- Live; actual systems
- Virtual; simulators
- Constructive; semi-automated forces

Live; instrumented vehicle  Virtual; flight simulator  Constructive; SAF
Utilities

Node of a distributed simulation system that performs a non-simulation support function

- Support data transport
- Log transmitted data “datalogger”
- Provide view into simulation “stealth viewer”
- Monitor and control overall execution “monitor”
Informal definitions

- **Interoperability**: the ability of models to meaningfully communicate in a distributed simulation
- **Composability**: the ability to combine and recombine models and model components into different complex simulations
Distributed simulation vs parallel simulation

- **Distributed simulation**
  - Nodes relatively loosely bound
  - Often (not always) real time
  - Usually implemented on separate computers connected via network and protocol

- **Parallel simulation**
  - Nodes relatively tightly bound
  - Often (not always) logical time
  - Usually implemented as multiple processes on single multi-processor computer

- Not disjoint
In depth:
Close Combat Tactical Trainer
Example distributed simulation: CCTT

- Close Combat Tactical Trainer
- Virtual simulators
  - Participants (trainees) inside vehicle simulators
  - Computer generated images of battlefield
- Combat training
  - Mounted and dismounted team tactics
  - Platoon to battalion units
- CCTT training sites
  - Fixed: U.S. x6, Germany, South Korea; 10-40 simulators per site
  - Mobile: x8; 1 simulator per trailer
CCTT simulator external views
CCTT simulator internal views

M1 commander

M1 gunner

M1 driver

M1 loader
CCTT simulator internal views

HMMWV driver

HMMWV observer

Dismounted Infantry

M2 driver
CCTT simulator out-the-window views
Additional CCTT images

- Enhanced M1 turret
- Battlemaster station
- Bridging and burning
Additional CCTT images

CCTT reconfigurable vehicle trainer

CCTT reconfigurable vehicle trainer
CCTT technical details

- Simulators connected via network
  - Distributed simulation, using DIS protocol
  - Other model types connected, e.g., logger, SAF
- Virtual terrain
  - High-fidelity geospecific or geotypical
  - 9x locations where potential engagements anticipated
  - Central Germany, Kosovo, Korea, Baghdad, NTC, …
Terrain

- **Terrain database**
  - Represents geographic area of scenario
  - Identical (or correlated) among linked models

- **Representation**
  - Terrain surface formed from polygons (triangles)
  - Texture, type associated with each triangle
  - $x, y, z$ values at vertices on 2D grid or arbitrary
  - Features (e.g., trees, buildings) separate
Introduction to interoperability protocols
Definition

**Distributed simulation protocol.** Network protocol designed to support a category of distributed simulation systems.

General protocol characteristics
- Definitions of
  - Data items
  - Message formats
  - Interaction sequences
- Standardized to support interoperability
Military distributed simulation protocols

- Simulator Networking SIMNET
  - First functional distributed simulation protocol
  - Homogenous, entity-level, mostly virtual
- Distributed Interactive Simulation Simulation DIS
  - Expanded capabilities w.r.t. SIMNET
  - Heterogeneous, entity-level, mostly virtual
- Aggregate Level Simulation Protocol ALSP
  - Heterogeneous logical time constructive
- High Level Architecture HLA
  - General purpose, subsumes previous protocols
- Test and Training Enabling Architecture TENA
  - Designed with test range applications in mind
Distributed simulation protocol development

SIMNET
- Virtual; real-time; entity level; 1980s

DIS
- Virtual; real-time; entity level; 1990s

HLA
- General purpose; 1995+

TENA
- Ranges; real-time; entity level; 2000s

ALSP
- Constructive; logical-time; aggregate level; 1990s
Distributed simulation protocol: Simulator Networking (SIMNET)

- **Characteristics**
  - Mounted combat
  - Distributed, virtual, entity level, real-time
  - Homogenous, proprietary
  - Both protocol and simulation system

- **Purpose:** team tactics training
Distributed simulation protocol: Aggregate Level Simulation Protocol (ALSP)

- Protocol designed to link constructive models
- Time management capabilities
  - Synchronize and advance simulation time, i.e., logical time
  - Simulation time different from wall-clock time
- Some models linked with ALSP
  - CBS (Corps Battle Simulation)
  - AWSIM (Air Warfare Simulation)
  - JTC (Joint Training Confederation)
Distributed Interactive Simulation
Distributed simulation protocol: Distributed Interactive Simulation (DIS) [IEEE, 1995]

• Development history
  ▪ Developed from SIMNET, beginning early 1990s
  ▪ Exploited lessons learned from SIMNET

• Characteristics
  ▪ Mounted combat
  ▪ Distributed, virtual, entity level, real-time
  ▪ Heterogeneous, non-proprietary
  ▪ Open protocol standard development process

• Used for multiple simulation systems
Basic concepts of DIS

- Simulation nodes
  - Multiple distributed simulators, simulations, utilities
  - Exchange messages via a network (LAN)
- Network messages
  - Conform to predefined standard protocol
  - Called Protocol Data Units (PDUs)
  - Transmitted broadcast (UDP/IP, TCP/IP)
- Message purposes
  - Report entity state (movement, status)
  - Mediate interactions between entities
  - Manage or control simulation execution
Main parts of DIS protocol [Loper, 1995]

- Data items to be passed
- Format of data items
  - e.g., int vs. float, value enumerations
- Grouping of data items into messages (PDUs)
- Conditions for sending PDUs
  - Specific to PDU type
- Processing to perform upon receiving PDUs
  - Specific to PDU type
- Key algorithms to be shared among nodes
  - e.g., dead reckoning
Most common DIS PDU types

- **Entity State**
  - Announce entity existence, location, movement, and appearance

- **Fire**
  - Announce that entity has fired a weapon
  - Important for rendering muzzle flashes

- **Detonation**
  - Announce that round has hit entity or terrain

- **Collision**
  - Exchanged between colliding entities
## DIS PDU types

<table>
<thead>
<tr>
<th>PDU Family/Type</th>
<th>DIS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Entity Information/Interaction</strong></td>
<td></td>
</tr>
<tr>
<td>Entity State</td>
<td>1278.1</td>
</tr>
<tr>
<td>Entity State Update</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Collision</td>
<td>1278.1</td>
</tr>
<tr>
<td>Collision-Elastic</td>
<td>1278.1a</td>
</tr>
<tr>
<td><strong>Warfare</strong></td>
<td></td>
</tr>
<tr>
<td>Fire</td>
<td>1278.1</td>
</tr>
<tr>
<td>Detonate</td>
<td>1278.1</td>
</tr>
<tr>
<td><strong>Simulation Management</strong></td>
<td></td>
</tr>
<tr>
<td>Create Entity</td>
<td>1278.1</td>
</tr>
<tr>
<td>Remove Entity</td>
<td>1278.1</td>
</tr>
<tr>
<td>Start/Resume</td>
<td>1278.1</td>
</tr>
<tr>
<td>Stop/Freeze</td>
<td>1278.1</td>
</tr>
<tr>
<td>Acknowledge</td>
<td>1278.1</td>
</tr>
<tr>
<td>Action Request</td>
<td>1278.1</td>
</tr>
<tr>
<td>Action Response</td>
<td>1278.1</td>
</tr>
<tr>
<td>Data Query</td>
<td>1278.1</td>
</tr>
<tr>
<td>Set Data</td>
<td>1278.1</td>
</tr>
<tr>
<td>Data</td>
<td>1278.1</td>
</tr>
<tr>
<td>Event Report</td>
<td>1278.1</td>
</tr>
<tr>
<td>Comment</td>
<td>1278.1</td>
</tr>
<tr>
<td><strong>Radio Communications</strong></td>
<td></td>
</tr>
<tr>
<td>Transmitter</td>
<td>1278.1</td>
</tr>
<tr>
<td>Signal</td>
<td>1278.1</td>
</tr>
<tr>
<td>Receiver</td>
<td>1278.1</td>
</tr>
<tr>
<td>Intercom Control</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Intercom Signal</td>
<td>1278.1a</td>
</tr>
<tr>
<td><strong>Distributed Emission Regeneration</strong></td>
<td></td>
</tr>
<tr>
<td>Electromagnetic Emission</td>
<td>1278.1</td>
</tr>
<tr>
<td>Designator</td>
<td>1278.1</td>
</tr>
<tr>
<td>IFF/ATC/NAVAIDS</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Underwater Acoustic</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Supplemental Emissions/Entity State</td>
<td>1278.1a</td>
</tr>
<tr>
<td><strong>Logistics Support</strong></td>
<td></td>
</tr>
<tr>
<td>Service Request</td>
<td>1278.1</td>
</tr>
<tr>
<td>Re-supply Offer</td>
<td>1278.1</td>
</tr>
<tr>
<td>Re-supply Received</td>
<td>1278.1</td>
</tr>
<tr>
<td>Re-supply Cancel</td>
<td>1278.1</td>
</tr>
<tr>
<td>Repair Complete</td>
<td>1278.1</td>
</tr>
<tr>
<td>Repair Response</td>
<td>1278.1</td>
</tr>
<tr>
<td><strong>Minefield</strong></td>
<td></td>
</tr>
<tr>
<td>Minefield State</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Minefield Query</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Minefield Data</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Minefield Response NACK</td>
<td>1278.1a</td>
</tr>
</tbody>
</table>

Parts 1 and 2 of 3
## PDU Family/Type

<table>
<thead>
<tr>
<th>Synthetic Environment</th>
<th>DIS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Environmental Process</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Gridded Data</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Point Object State</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Linear Object State</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Areal Object State</td>
<td>1278.1a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Entity Management</th>
<th>DIS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aggregate State</td>
<td>1278.1a</td>
</tr>
<tr>
<td>IsGroupOf</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Transfer Control Request</td>
<td>1278.1a</td>
</tr>
<tr>
<td>IsPartOf</td>
<td>1278.1a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Live Entity</th>
<th>DIS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time Space Position Information</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Appearance</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Articulated Parts</td>
<td>1278.1a</td>
</tr>
<tr>
<td>LE Fire</td>
<td>1278.1a</td>
</tr>
<tr>
<td>LE Detonation</td>
<td>1278.1a</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-Real Time</th>
<th>DIS Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>Action Request</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Action Response</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Set Data</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Data Query</td>
<td>1278.1a</td>
</tr>
<tr>
<td>Data</td>
<td>1278.1a</td>
</tr>
</tbody>
</table>
Example DIS interaction: direct fire

- Entities interact by exchanging PDUs
- Protocol defines PDU sequence for interaction
Example DIS direct fire interaction sequence

1. SAF system $T$ sends Entity State PDU for T-72 $t$ at a specific location.
2. Simulator $S$ receives Entity State PDU for T-72 $t$ and uses it to render $t$ in view port of M1 $s$.
3. The crew of M1 $s$ see T-72 $t$ and fire on it in simulator $S$.
4. Simulator $S$ sends a Fire PDU announcing that $s$ has fired.

Continued on next slide
5. Simulator $S$ determines if a hit was scored. (Assume yes.)
6. Simulator $S$ sends a Detonation PDU to SAF system $T$ announcing that $t$ was hit by $s$; PDU includes details of hit.
7. SAF system $T$ determines effect of hit on $t$. (Assume $t$ is destroyed.)
8. SAF system $T$ sends a new Entity State PDU for $t$ with its new appearance (destroyed).
DIS and local copies

- **Definition**
  - Remote nodes send entity data updates at intervals
  - Local nodes keep copies to use between updates
  - Non-persistent form of redundancy
  - Motivated by performance (vs network query)

- **Example uses of local copies**
  - Sensing and detection (e.g., intervisibility)
  - Combat (e.g., direct fire)

- **Local copies updated via dead reckoning**
Example DIS common algorithm: dead reckoning [Lin, 1995]

- **Purpose**
  - Reduce network message volume
  - Provide remote entity location whenever needed

- **Mechanism**
  - Node projects (“dead reckons”) location of each remote entity since last ES PDU received
  - Nodes both model and dead reckon local entities, send ES PDU when discrepancy too large
Dead reckoning sequence

1. T broadcasts Entity State PDU for t with position and velocities
2. S receives Entity State PDU for t, sets position and velocities
3. S and T both dead reckon t over time
4. T models actual movement of t over time concurrent
5. T finds that discrepancy between actual and dead reckoned positions for t exceeds threshold
6. T broadcasts Entity State PDU for t with position and velocities
7. S receives Entity State PDU for t, set position and velocities
Example DIS-based Dist Sim system: Close Combat Tactical Trainer

- Models mounted combat
- Virtual, entity level, real-time
- Heterogenous, non-proprietary
- Includes simulators, SAF, utilities
- Uses DIS protocol
Example DIS-based Dist Sim system: Distributed Mission Training (DMT)

- U. S. Air Force flight simulation system
- Used for training: mission and team, not skill
- Distributed sim, connected via DIS and/or HLA
DMT pilot station [Boeing, 2004]

- Simulated cockpit for aircraft pilot
  - F-15C and other aircraft
  - Includes flight dynamics, controls, visuals
- Real-time, virtual environment for trainee
Threat station

- Control friendly/threat aircraft
  - Simple simulated cockpit
  - Used for operator, i.e., non-trainee
- Produce specific aircraft behaviors for exercise
Tactical environment generator

- Simulates mission environment
  - Constructive simulations; SAF for exercise
  - Friendly and threat entities and sensors
- Generates context for training
Instructor/operator station

- Simulation control capabilities
  - Construct scenarios
  - Stop, start, replay, intervene in exercise
- Control and monitor exercise
Brief/debrief station

- Briefing and visualization capabilities
  - Map overview of mission/exercise
  - Pilot-generated mission/exercise planning
- Preparation and after-action review
High Level Architecture
Distributed simulation protocol: High Level Architecture (HLA)

HLA is a general purpose distributed simulation protocol and architecture.

“The High Level Architecture is an architecture for reuse and interoperability of simulations.”

[Dahmann, 1998a]
“Major functional elements, interfaces, and design rules, pertaining as feasible to all DOD simulation applications, and providing a common framework within which specific simulation system architectures can be defined.”  [DOD, 1996] [DOD, 1998]
Design premises of HLA [Dahmann, 1998b]

- **Architecture**
  - Distributed simulation systems assembled by connecting nodes via network and protocol

- **Flexibility**
  - No fixed protocol can serve all users’ needs, nor can all future applications be anticipated
  - Protocol must allow customization

- **Separation of functionality**
  - Application-specific (i.e., data definition)
  - General infrastructure (i.e., data transport)
HLA development history

- **Initial development** [Dahmann, 1998b]
  - Initiated 1995
  - Sponsored and organized by DMSO
  - Design and implementation overseen by Architecture Management Group

- **Proto-Federations**
  - Test implementations of HLA federations 1996
  - Four proto-federations, different applications: Analysis, Engineering, Joint Training, Platform
  - Each had multiple federates
  - Many lessons [Harkrider, 1996a] [Harkrider, 1996b] [Harkrider, 1997]
Subsequent developments
- HLA designated DoD distributed simulation standard ("Kaminski Mandate") 1996
- DoD 1.3 version made DoD standard 1998
- IEEE 1516 version made IEEE standard 2000

Current status
- Revision to IEEE 1516 standard “HLA Evolved”, made standard 2010 [IEEE, 2010a] [IEEE, 2010b] [IEEE, 2010c]
- HLA widely adopted, extensively used
- Many federates, federations, tools developed
HLA specifications

- **Rules** [IEEE, 2010a]
- **Object Model Template** [IEEE, 2010c]
- **Interface Specification** [IEEE, 2010b]
HLA terms

- **Federate**: individual node in distributed simulation system (simulation or utility)
- **Federation**: set of interoperating nodes
• **Object Model**: specification of data to be exchanged by a federation
• **Run-Time Infrastructure (RTI)**: software that supports exchange of data in federation
• **RTI service**: specific capability provided by RTI
Federates

Support Utilities

Simulations

Live Participants

Live Player Interfaces

Interface

Run-Time Infrastructure

Federation Management
Object Management
Time Management

Declaration Management
Ownership Management
Data Distribution Management
HLA Rules

- Define responsibilities and restrictions
- 10 rules total
- 5 rules each for federates and federations
Federation rules [IEEE, 2010a]

1) Federations shall have an HLA FOM, documented in accordance with the HLA OMT.
2) In a federation, all simulation-associated object instance representation shall be in the federates, not in the RTI.
3) During a federation execution, all exchange of FOM data among joined federates shall occur via the RTI.
4) During a federation execution, joined federates shall interact with the RTI in accordance with the HLA interface specification.
5) During a federation execution, an instance attribute shall be owned by at most one joined federate at any given time.
Federate rules [IEEE, 2010a]

6) Federates shall have an HLA Simulation Object Model (SOM), documented in accordance with the HLA OMT.
7) Federates shall be able to update and/or reflect any instance attributes and send and/or receive interactions, as specified in their SOMs.
8) Federates shall be able to transfer and/or accept ownership of instance attributes dynamically during a federation execution, as specified in their SOMs.
9) Federates shall be able to vary the conditions (e.g., thresholds) under which they provide updates of instance attributes, as specified in their SOMs.
10) Federates shall be able to manage local time in a way that will allow them to coordinate data exchange with other members of a federation.
HLA object models

- Define federation data to be exchanged
- Object classes and attributes
  - Persistent objects
  - Hierarchy, single inheritance
- Interaction classes and parameters
  - Non-persistent interactions between objects
  - Hierarchy, single inheritance
- Documented per Object Model Template
- Similar, not same, as “object-oriented”
HLA object models

- Define federation data to be exchanged
- Object classes and attributes
  - Persistent objects
  - Hierarchy, single inheritance
- Interaction classes and parameters
  - Non-persistent interactions between objects
  - Hierarchy, single inheritance
- Documented per Object Model Template
- Similar, not same, as “object-oriented”
### Example OM: Class table

<table>
<thead>
<tr>
<th>HLA object Root (N)</th>
<th>Employee (N)</th>
<th>Food (S)</th>
<th>Entree (S)</th>
<th>SideDish (S)</th>
<th>Dessert (S)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Customer (P)</td>
<td>Bill (P)</td>
<td>Order (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cook (P)</td>
<td>Cashier (P)</td>
<td>Dishwasher (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Greet (P)</td>
<td>Water (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Coffee (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Soda (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Soup (S)</td>
<td>ClamChowder (P)</td>
<td>Manhattan (P)</td>
<td>NewEngland (P)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Manhattan (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Appetizers (S)</td>
<td>Nachos (P)</td>
<td>BeefBouley (P)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Beef (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Chicken (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Seafood (S)</td>
<td>Fish (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Shrimp (P)</td>
<td>Lobster *[Note1] (P)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Lobster *[Note2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Pasta (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Corn (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Broccoli (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>BakedPotato (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Cake (P)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>IceCream (S)</td>
<td>Chocolate (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Vanilla (P)</td>
</tr>
</tbody>
</table>

[IEEE, 2010b]
Example OM: Attribute table

<table>
<thead>
<tr>
<th>Object</th>
<th>Attribute</th>
<th>Datatype</th>
<th>Update type</th>
<th>Update condition</th>
<th>D/A</th>
<th>P/S</th>
<th>Available dimensions</th>
<th>Transportation</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>HLAObject Root</td>
<td>HLAToken</td>
<td>NA</td>
<td>NA</td>
<td></td>
<td>N</td>
<td>N</td>
<td>NA</td>
<td>HLAAvailable</td>
<td>Time-stamp</td>
</tr>
<tr>
<td>Employee</td>
<td>PayRate</td>
<td>Dollar</td>
<td>Conditional</td>
<td>Monthly increase</td>
<td>N</td>
<td>N</td>
<td>NA</td>
<td>HLAAvailable</td>
<td>Time-stamp</td>
</tr>
<tr>
<td></td>
<td>YearsOfService</td>
<td>Periodic</td>
<td>1/year</td>
<td>DA</td>
<td></td>
<td></td>
<td>NA</td>
<td>Time-stamp</td>
<td></td>
</tr>
<tr>
<td>Home Number</td>
<td>HLAASCII string</td>
<td>Conditional</td>
<td>Employee request</td>
<td>DA</td>
<td></td>
<td>NA</td>
<td>Time-stamp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Home Address</td>
<td>AddressType</td>
<td>Conditional</td>
<td>Employee request</td>
<td>DA</td>
<td></td>
<td>NA</td>
<td>Time-stamp</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Employee Worker</td>
<td>Efficiency</td>
<td>Wt/Mv</td>
<td>Conditional</td>
<td>Performance review</td>
<td>DA</td>
<td></td>
<td>NA</td>
<td>HLAAvailable</td>
<td>Time-stamp</td>
</tr>
<tr>
<td></td>
<td>Cheerfulness</td>
<td>Wt/Mv</td>
<td>Conditional</td>
<td>Performance review</td>
<td>DA</td>
<td></td>
<td>NA</td>
<td>Time-stamp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>State</td>
<td>Wt/Tsk</td>
<td>Conditional</td>
<td>Work flow</td>
<td>DA</td>
<td></td>
<td>NA</td>
<td>Time-stamp</td>
<td></td>
</tr>
<tr>
<td>Food/Drink</td>
<td>NumberCup</td>
<td>Drink</td>
<td>Conditional</td>
<td>Customer request</td>
<td>N</td>
<td></td>
<td>BarQuantity</td>
<td>HLAAvailable</td>
<td>Time-stamp</td>
</tr>
<tr>
<td></td>
<td>DrinkCount</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Time-stamp</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Flavor</td>
<td>FlvType</td>
<td>Conditional</td>
<td>Customer request</td>
<td>N</td>
<td></td>
<td>SodaFlavor</td>
<td>HLAAvailable</td>
<td>Time-stamp</td>
</tr>
</tbody>
</table>

Note: NA

[IEEE, 2010b]
### Example OM: Interaction and Parameter tables

<table>
<thead>
<tr>
<th>Interaction</th>
<th>Parameter</th>
<th>Datatype</th>
<th>Available dimensions</th>
<th>Transportation</th>
<th>Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>CustomerSeated</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>HLA reliable</td>
<td>TimeStamp</td>
</tr>
<tr>
<td>FoodServed.MainCourseServed</td>
<td>TemperatureOk</td>
<td>ServiceStat</td>
<td>WaiterId</td>
<td>HLA reliable</td>
<td>TimeStamp</td>
</tr>
<tr>
<td></td>
<td>AccuracyOk</td>
<td>ServiceStat</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>TimelinessOk</td>
<td>HLA boolean</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Note**: NA

[IEEE, 2010b]
HLA object attributes

- Redundancy?
  - Multiple objects (instances) of a single object class may be created (instantiated) during execution
  - Same set of data items (attributes) for each object, but different objects, attributes, values
  - No redundancy

- Inconsistency?
  - Ownership (right to update) of an object attribute may be transferred from federate to federate
  - Ownership limited by HLA to one federate at a time
  - No inconsistency
HLA Example DIF

<?xml version="1.0"?>
<!DOCTYPE objectModel SYSTEM "HLA.dtd">
<objectModel
  DTVersion="16.0.2"
  name="Example"
  type="POM"
  version="1.0"
  date="2010-06-01"
  purpose="Provide an example of an HLA POM"
  sponsor="UH"
>
  <objects>
    <objectClass name="HLAObjectRoot"
      sharing="Neither">
      <attribute name="HLAprivilegeLevel"objectClass="controller"
        dataType="NA"
        updateType="NA"
        updateCondition="NA"
        ownership="NOTransfer"
        sharing="Neither"
        dimensions="NA"
        transportation="HLAAvailable"
        order="TimeStamp"/>
    </objectClass>

    <objectClass name="UserBaseClass"
      sharing="Neither"
      semantic="This object class is the base of all user-defined object classes">
      <objectClass name="UserSubclass"
        sharing="PublishSubscribe"
        semantic="This is a subclass of UserBaseClass">
        <attribute name="UserAttribute"
          dataType="UserDatatype"
          updateType="Static"
          updateCondition="NA"
          ownership="NOTransfer"
          sharing="PublishSubscribe"
          dimensions="NA"
          transportation="HLAAvailable"
          order="timeStamp"
          semantic="Attribute of UserSubclass"/>
      </objectClass>
    </objectClass>
  </objects>
</objectModel>
Special HLA object models

• Federation object model (FOM)
  ▪ Shared object model in federation
  ▪ Objects and interactions

• Simulation object model (SOM)
  ▪ Object model for single federate
  ▪ Objects and interactions
  ▪ External only

• Federation’s FOM generally a subset of the union of the federates’ SOMs

• RPR FOM recreates DIS as HLA OM
HLA Interface Specification

• Purpose
  ▪ Formal definition of operations ("services") used to exchange simulation and control information in a federation execution
  ▪ Formal specification of interface between RTI and federates, defined as a set of functions with API

• Interface Specification and the RTI
  ▪ Interface Spec; defines services and software interface to use them
  ▪ RTI; implements and executes the services
## Concepts and Protocols

<table>
<thead>
<tr>
<th>Service Category</th>
<th>Functionality</th>
<th>Services 1516-2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Federation Management</td>
<td>Create, control, destroy federation executions</td>
<td>31</td>
</tr>
<tr>
<td></td>
<td>Join and resign federation executions</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Pause, resume, checkpoint, restart</td>
<td></td>
</tr>
<tr>
<td>Declaration Management</td>
<td>Announce intent to send or receive object and interaction information</td>
<td>12</td>
</tr>
<tr>
<td>Object Management</td>
<td>Create and delete objects</td>
<td>29</td>
</tr>
<tr>
<td></td>
<td>Send and receive object attribute updates</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Send and receive interactions</td>
<td></td>
</tr>
<tr>
<td>Ownership Management</td>
<td>Transfer ownership of object attributes between federates</td>
<td>18</td>
</tr>
<tr>
<td>Time Management</td>
<td>Control and synchronize simulation time</td>
<td>23</td>
</tr>
<tr>
<td>Data Distribution Management</td>
<td>Filter data sent between federates</td>
<td>12</td>
</tr>
<tr>
<td>Support</td>
<td>Provide infrastructure status information to federates</td>
<td>43</td>
</tr>
</tbody>
</table>
Federation Management services [IEEE, 2010c]

- **Purpose**
  - Control federation executions
  - Join and resign from federation executions

- **Example services**
  - Create Federation Execution
  - Join Federation Execution
  - Resign Federation Execution
  - Destroy Federation Execution
  - Request Federation Save
  - Federation Saved †
Declaration Management services [IEEE, 2010c]

• **Purpose**
  ▪ Announce intent to send or receive data
  ▪ Based on object and interaction classes

• **Example services**
  ▪ Publish Object Class Attributes
  ▪ Subscribe Object Class Attributes
  ▪ Publish Interaction Class
  ▪ Subscribe Interaction Class
Object Management services [IEEE, 2010c]

- **Purpose**
  - Create and delete objects
  - Send and receive object updates & interactions

- **Example services**
  - Register Object Instance
  - Discover Object Instance †
  - Update Attribute Values
  - Reflect Attribute Values †
  - Delete Object Instance
  - Remove Object Instance †
Ownership Management services [IEEE, 2010c]

- **Purpose**
  - Transfer ownership of object attributes between federates

- **Example services**
  - Negotiated Attribute Ownership Divestiture
  - Request Attribute Ownership Assumption †
Time Management services [IEEE, 2010c]

• Purpose
  ▪ Control and synchronize simulation time

• Example services
  ▪ Time Advance Request
  ▪ Time Advance Grant †
  ▪ Retract
  ▪ Request Retraction †
Data Distribution Management services [IEEE, 2010c]

- **Purpose**
  - Filter data sent between federates
  - Based on data value ranges

- **Example services**
  - Create Region
  - Register Object Instance With Regions
  - Subscribe Object Class Attributes With Regions
Support services [IEEE, 2010c]

• **Purpose**
  - Provide infrastructure status information

• **Example services**
  - Get Object Class Handle
  - Get Attribute Handle
  - Enable Callbacks
  - Disable Callbacks
Run-Time Infrastructure (RTI)

- Not a part of the definition of HLA
- Software realization of the HLA definition
- Provides run-time support to federation
  - Transports data between federates
  - Controls federation execution
  - Manages simulation time
Logical view of a federation

- Federates send data to and receive data from RTI, via services
- RTI is intermediary between federates
Technical view of a federation

- LRC integrated into each federate
- Federate passes data to/from LRC via services; LRCs exchange data via network
- CRC handles special services

LRC = Local RTI Component  CRC = Central RTI Component
RTI services invoked via Ambassadors
HLA standards

- **DoD 1.3**
  - Original HLA standard
  - Initial RTI and HLA software implemented in DoD 1.3
  - DoD 1.3 software no longer supported
  - No longer in use?

- **IEEE 1516-2000**
  - Developed from DoD 1.3
  - Many improvements [DMSO, 2004] [Morse, 2004c]
  - Widely used
  - Federates, federations, tools, products available

- **HLA 1516-2010**
  - Developed from IEEE 1516
  - Standardized 2010 [IEEE, 2010a] [IEEE, 2010b] [IEEE, 2010c]
HLA compliance

- **Compliance**
  - Independent agencies test HLA federates
  - Successful test certifies “HLA compliance”
  - 1996 mandate connected compliance to funding

- **Approaches to compliance**
  - Native; use HLA directly
  - Middleware; HLA hidden in software layer
  - Gateway; protocol translator [Wood, 1999]

- **Non-compliance**
  - Waivers available, compliance less important
  - Non-compliant, non-standard implementations exist, e.g., JFCOM RTI-s
HLA summary

• Goals and premises
  ▪ General-purpose, flexible distributed simulation
  ▪ Architecture, protocol, middleware
  ▪ Provides data transport, other services
  ▪ Semantics in object models, not in HLA

• Defined by specifications
  ▪ Federates and federation follow Rules
  ▪ Data defined per Object Model Template
  ▪ Interface Specification defines services, interfaces

• Run-Time Infrastructure
  ▪ Implements services, transports data
  ▪ Multiple RTI versions available
Standards and compliance

- HLA standards (chronological order):
  - Independent testing for protocol compliance

Well established

- Numerous HLA federates, federations, applications
- Mature IEEE standard
Test and Training
Enabling Architecture
Distributed simulation protocol: Test and Training Enabling Architecture (TENA) [TENA, 2008]

- Designed for range (test, training) applications
  - Entity level
  - Live, virtual, constructive
  - Real-time response
- Protocol and architecture
  - Protocol; object model, messages
  - Architecture; common middleware
- Goals
  - Iterative improvement based on user feedback
  - Interoperability, reusability, composability
TENA components

- Architecture components
  - TENA Object Model
  - TENA Middleware
- Common context
  - Representation of the environment (SEDRIS)
- Software development support
  - Development process (TENA Technical Process)
  - Reusable tools (object model utilities)
  - Repository (software components)
  - Data archive (execution data)
TENA architecture overview

Constructive model  Live radar or telemetry  Hardware-in-the-loop  Execution monitor  AAR

- Range Resource Application
- Range Resource Application
- Range Resource Application
- Monitoring Application
- Analysis & Review Application

TENA Middleware

© 2013 University of Alabama in Huntsville; © 2013 Mikel D. Petty, Ph.D.
TENA application architecture
TENA Logical Range Object Model (LROM)

- Object definitions used in logical range execution
  - Contain objects needed in execution
  - LROM may come from multiple sources
- Common object model
  - Shared by all applications in logical range
  - Provides “common language”
- Incrementally enhanced
  - User developed objects, custom for logical range
  - Supported objects, available for reuse
  - Standard objects, approved for standardization
  - Latter expected to increase over time
TENA standard object models

- TENA-Radar-v2
- TENA-GPS-v1
- TENA-Platform
  - TENA-Platform-v3.1
  - TENA-PlatformDetails-v3
  - TENA-Affiliation-v1
  - TENA-UniqueID-v2
  - TENA-PlatformType-v1
  - DIS-EntityType-v2
  - TENA-Munition-v2.1
  - TENA-Engagement-v3.1
  - TENA-Organization-v1
  - TENA-EmbeddedSystem-v2
  - TENA-EmbeddedSensor-v21
  - TENA-EmbeddedWeapon-v2
- TENA-TSPI
  - TENA-TSPI-v4
  - TENA-Time-v1.1
  - TENA-Position-v1
  - TENA-Velocity-v1
  - TENA-Acceleration-v1
  - TENA-Orientation-v1
  - TENA-AngularVelocity-v1
  - TENA-AngularAcceleration-v1
  - TENA-ORM-v1
  - TENA-SRF-v1
  - TENA-SRFserver-v1
- TENA-AMO
  - TENA-AMO-v1
TENA standard object model example
TENA code generator

- Generates code for TENA Middleware
  - Input: TDL (TENA Description Language)
  - Output: C++
- Accessed via web front end
- Promotes interoperability via homogeneity
  - Same source code for every TENA application
  - Common algorithms embedded in generated code, e.g., coordinate conversion, unit conversion, data marshalling/demarshalling
TENA logical range example: InterTEC

- Large distributed LVC C4I Link-16 test
- TENA used for
  - Distribution of instrumentation data
  - Test control
  - Distributed simulation across multiple sites

10 locations
12 different applications
56 instances of applications
TENA summary

- Protocol and architecture designed for ranges
- Common middleware provides data transport
- Standard object models at entity level and below
- Numerous applications since development
Implementing interoperability
Achieving interoperability

- **Native**
  - Federate/application code uses protocol directly
  - e.g., most DIS applications

- **Middleware**
  - Common software layer used by all federates
  - Application uses protocol indirectly via middleware
  - e.g., TENA, arguably HLA

- **Gateway**
  - Application uses one protocol, e.g., DIS
  - Gateway translates application protocol to another, e.g., DIS/HLA gateway
Concepts and Protocols

General interoperability issues

• Protocol compliance
  ▪ Properly formatted data items and messages
  ▪ Correct send and receive processing

• Model consistency issues
  ▪ Terrain correlation
  ▪ Consistent weapons effects
  ▪ Consistent object models

• Technical issues
  ▪ Coordinate systems and coordinate conversions
  ▪ Byte alignment and endianness
  ▪ Ambiguities in protocol standard
Interoperability issue: model consistency

Model A
Free fall without air resistance

\[ h(t) = -4.9t^2 + vt + s \]

Model B
Free fall with air resistance

\[ h(t) = -4.9t^2 + vt + s + r(t) \]

\( r(t) \) is a notional model of air resistance
Example interoperability implementation: 1992 I/ITSEC DIS Interoperability Demonstration

• First large scale use demonstration of DIS
  ▪ 39 heterogeneous simulations and utilities
  ▪ Demonstrated on I/ITSEC exhibit floor
• Testing
  ▪ IST CGF used as protocol “gold standard” [Loper, 1993]
  ▪ Each simulation tested for interoperability against it
  ▪ 7 days, 24 hours, 12 hour shifts
• Demonstration
  ▪ Network cabling crushed; replaced after midnight
  ▪ Backup demo scripted in IST CGF system, not used
  ▪ Demo successful
1992 I/ITSEC DIS Interoperability Demo, testing
1992 I/ITSEC DIS Interoperability Demo, event

- Replacing cabling
- Developing backup
- Out-the-window
- Post-event
1992 I/ITSEC DIS Interoperability Demo, *Wired*

“The demo’d the new standard on a network link-up I/ITSEC … live.

The had to rip up some of the Ethernet wiring that they’d laid before the show because it had so many crimp-failures …

It got hairy for a while there. But they got the demo to run. The protocol worked just fine.

There was some interesting stuff backstage …

There was a handscrawled brag on a backstage chalkboard, written by the techies from Orlando: “DIS Interoperability Demonstration. Today’s feature: DIS. Tomorrow: the holodeck.”

[Sterling, 1993]
In depth:
Terrain issues
in distributed simulation
Persistent data in IT

Entity (Customer)

Representation

Data set

Input/output

Application program

During execution  Between executions  During execution

Time

© 2013 University of Alabama in Huntsville; © 2013 Mikel D. Petty, Ph.D.
Non-persistent data in M&S

- Entity (Tank)
- Representation
- Data set
- Input/output

During simulation  Between simulations  During simulation

Time

© 2013 University of Alabama in Huntsville; © 2013 Mikel D. Petty, Ph.D.
Terrain representation and terrain databases

- **Terrain representation**
  - Polygons (usually triangles) form surface of earth
  - Texture, type associated with each polygon
  - Features (e.g., trees, buildings) located on polygons

- **Terrain database**
  - Data set containing terrain data (e.g., vertex $x$, $y$, $z$ coordinates)
  - Several standard formats (e.g., CTDB)
Terrain images: Rural

- Surface polygons may have digital photos as textures
- Used for visualization, not simulation
Terrain images: Urban
Terrain alternative 1: integrated

- Single copy of terrain database at single nodes
- Other nodes request terrain data via queries
Terrain alternative 2: replicated

- Each node has its own copy of the terrain database
- No network queries required to retrieve terrain data
- Resulting issues: terrain correlation, dynamic terrain
Terrain correlation issue: floating entities

- Inconsistent terrain database heights [Schiavone, 1995]
- Causes ground entities to “float”
Terrain correlation issue: fair fight

- Inconsistent terrain database features  [Petty, 1996]
- Causes “fair fight” problems
Dynamic terrain

• Definition
  ▪ Some simuland actions can change terrain
  ▪ Changes must be applied to all replicated TDBs

• Examples
  ▪ Bulldozer digs entrenchment
  ▪ Bomb creates crater
Dynamic terrain issues

- Describing terrain changes in the protocol
- Algorithms for inserting terrain changes into local TDBs
- Time required to generate, send, receive, apply updates
Summary and references
Concepts and Protocols summary

- Distributed simulation
  - Architecture for networking simulations
  - Combines simulations and support nodes
- Interoperability protocols
  - Define data content, data transport, interaction sequences
  - Some require specific simulation architectures
  - Military examples: SIMNET, DIS, ALSP, HLA, TENA
- Related issues
  - Implementing interoperability
  - Terrain representation and correlation
  - Semi-automated forces
References


End