Live-Virtual-Constructive Interoperability Techniques

Randy Saunders & Ed Powell

30 November 2018

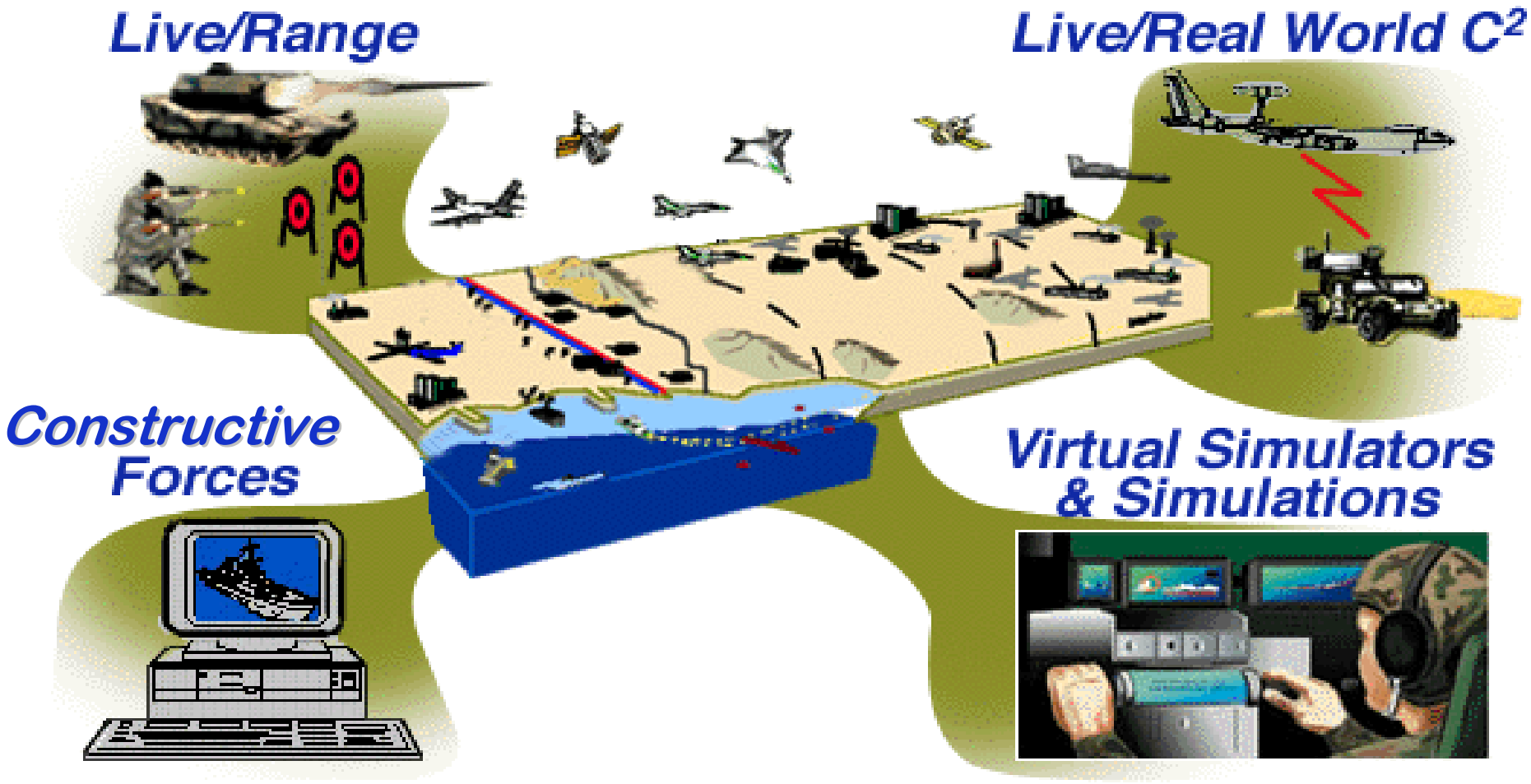
Agenda

0800-0815	Introduction	Saunders
0815-0840	LVC Integration – The Issues	Powell
0840-0900	DIS	Powell
0900-1000	HLA	Saunders
1000-1015	break	
1015-1115	TENA	Powell
1115-1145	Future Possibilities	Saunders
1145-1200	Summary/Recommendations	

Introduction

- What is Live-Virtual-Constructive (LVC) Interoperability?
  - Why should you want it?
  - What's the problem?

LVC Interoperability



Why is LVC Interoperability Important?

- Limitations on Live Opportunities
  - High cost for a large force, both manpower and equipment.
  - Joint operation requires more players, OOTW even more so.
  - Environmental factors are beyond control.
- Limitations on Virtual Simulation
  - High cost for a large force, both manpower and equipment.
  - Joint operation requires more players, OOTW even more so.
  - Verification and Validation efforts required to establish confidence.
- Limitations on Constructive Simulation
  - Verification and Validation efforts required to establish confidence.
  - Representation of human behavior is beyond the current state-of-the-art in computer science.

The LVC Architecture Issue

- Current LVC environments are not inherently interoperable.
  - **High Level Architecture** (HLA) **and Distributed Interactive Simulation** (DIS) **are most often used for integrating virtual and constructive assets,**
  - **Test & Training Enabling Architecture** (TENA) **is widely used in testing and to integrate live assets into exercises/events.**
  - **Common Training Instrumentation Architecture** (CTIA) **promotes commonality among the U.S. Army's instrumented ranges and home stations; LVC - Integrated Architecture** (LVC-IA) **is next-generation Army multi-echelon, integrated, joint, training and mission rehearsal environment;**
- Multiple protocols, gateways, and object models are often used to bring an LVC Environment together.
  - **Interoperability and efficiency issues arise when bringing disparate protocols and entities together in a common operational environment.**
  - **Complexity, disconnects, duplication of effort, risk, and costs increase with multiple architectures.**

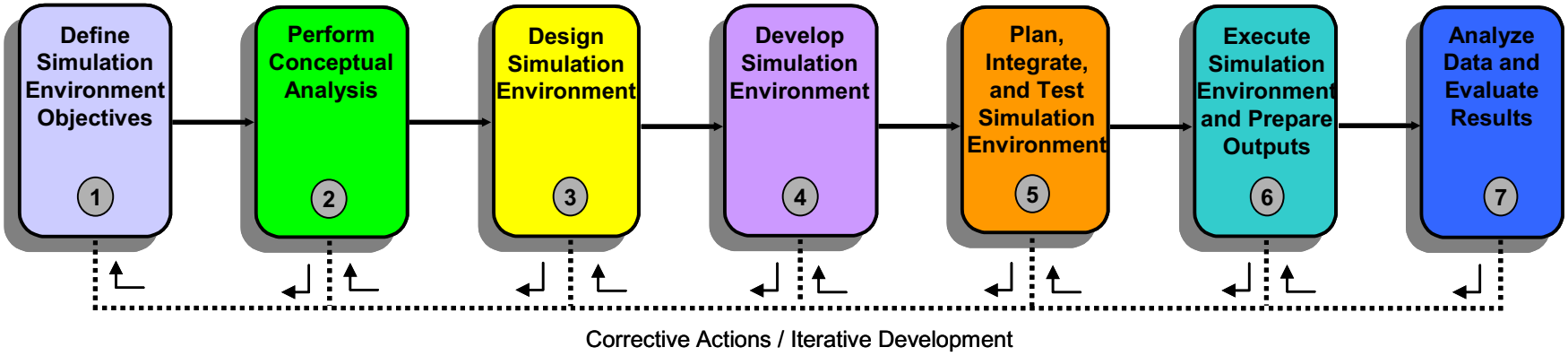
At least four communities agree; critical review needed to develop way forward for efficient, effective interoperability.

A Framework for Interoperability

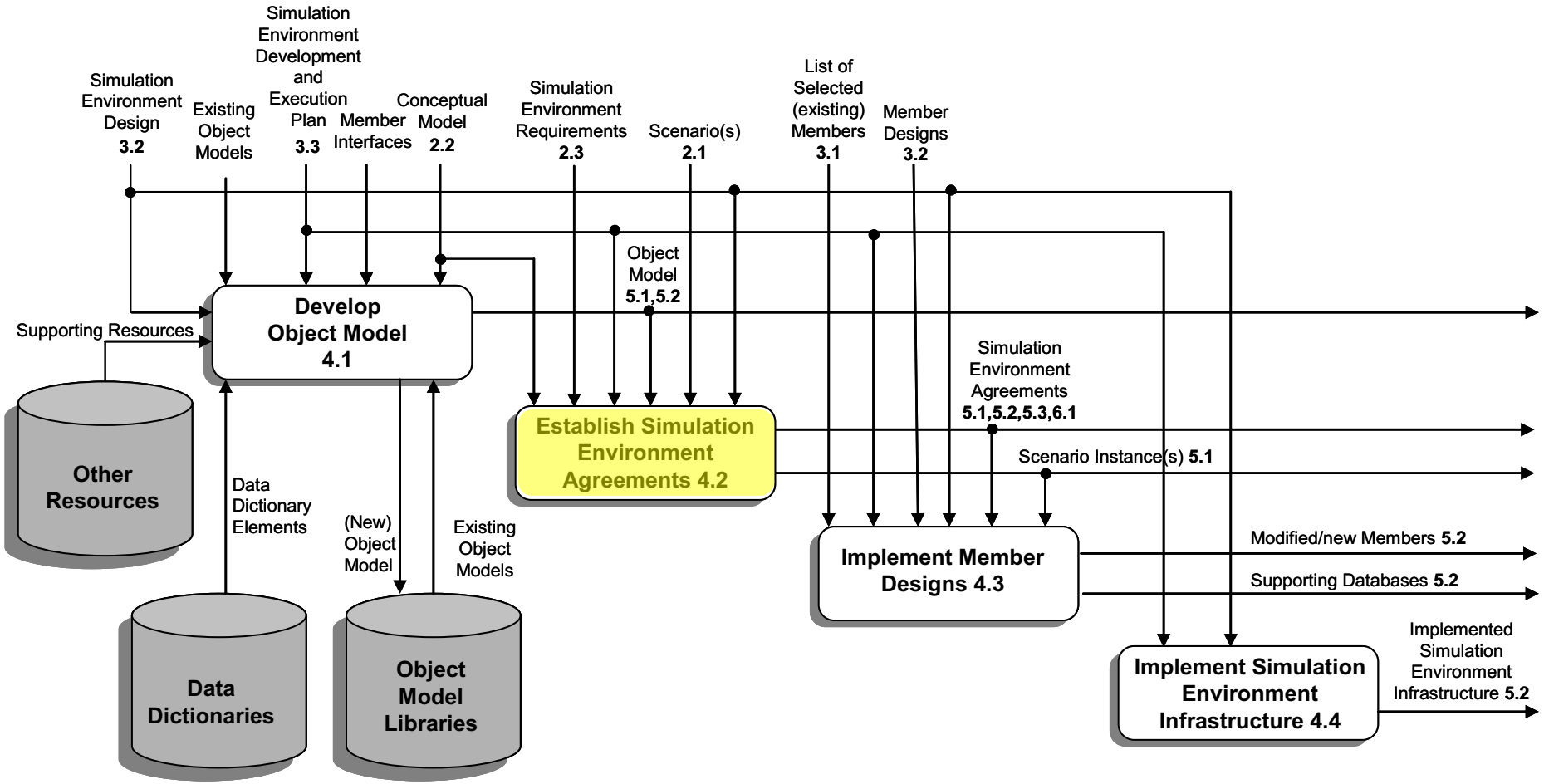
- Technical
  - Define an architecture (or set of integrated architectures) that can support the coherent runtime exchange of information among cooperating simulations
- Process
  - Technical - Define a robust technical operations model that identifies the time-sequenced set of activities and tasks necessary to achieve a desired level of interoperability
  - Business - Create an effective, efficient business operations model to ensure the availability of supporting software/data infrastructure
- Standards
  - Define opportunities for architecture, process, software, and data standards to facilitate cooperative development

Distributed Simulation Engineering and Execution Process (DSEEP)

- New SISO/IEEE-sponsored initiative to define a common systems engineering approach to the development of distributed simulation environments (IEEE P1730)
- Defines a standard systems engineering methodology for all distributed simulation users
- Includes architecture-specific views of the standard process
  - DIS, HLA, TENA
- Initially based on HLA FEDEP, but working group includes a very wide user base
  - All affected architectures
  - International participants
- Multi-Architecture Overlay (DMAO) developed to address LVC concerns



DSEEP (Step 4 Expanded)



Federation Engineering Agreements Template (FEAT)

- SISO recently finalized a standardized XML template for representing federation agreements.
- Existing agreements from a dozen large, multi-architecture federations were examined and included in the schema.
- An initial tool for editing an agreements document in the standard form has been produced and is available as open source at: http://sourceforge.net/p/feateditor/

Architectural Implications

- DSEEP encourages engineering in a loosely-coupled architecture
  - The Design (3), Develop (4), and Integrate (5) steps can be done in an iterative manner.
- Technical characteristics to examine
  - Distributed architectures exploit additional computers while using the network to reduce coupling
  - Existing simulations can be reused or adapted to new scenarios
  - Connections can bridge the differences between simulations

Simulation Integration: Live, Virtual, and Constructive (LVC) What this means and why?

Dr. Edward T. Powell

epowell@tena-sda.org

Terms

- **A "model" is a *simplified mathematical representation* of a real-world object**
  - A selective re-creation of reality based on the creator's objectives and evaluations as to which aspects of reality are *important* to the *purpose* of the model
  - Two *completely different* models of a given system may both be right, because they serve different objectives

- **A "simulation" is a mechanism for evolving a model or set of models *over time***

Model

Reality

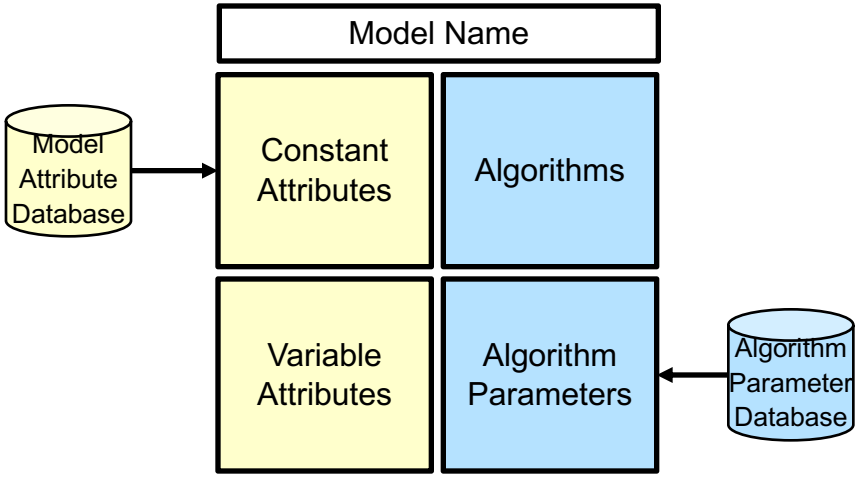
Objectives

Purpose

Evaluation

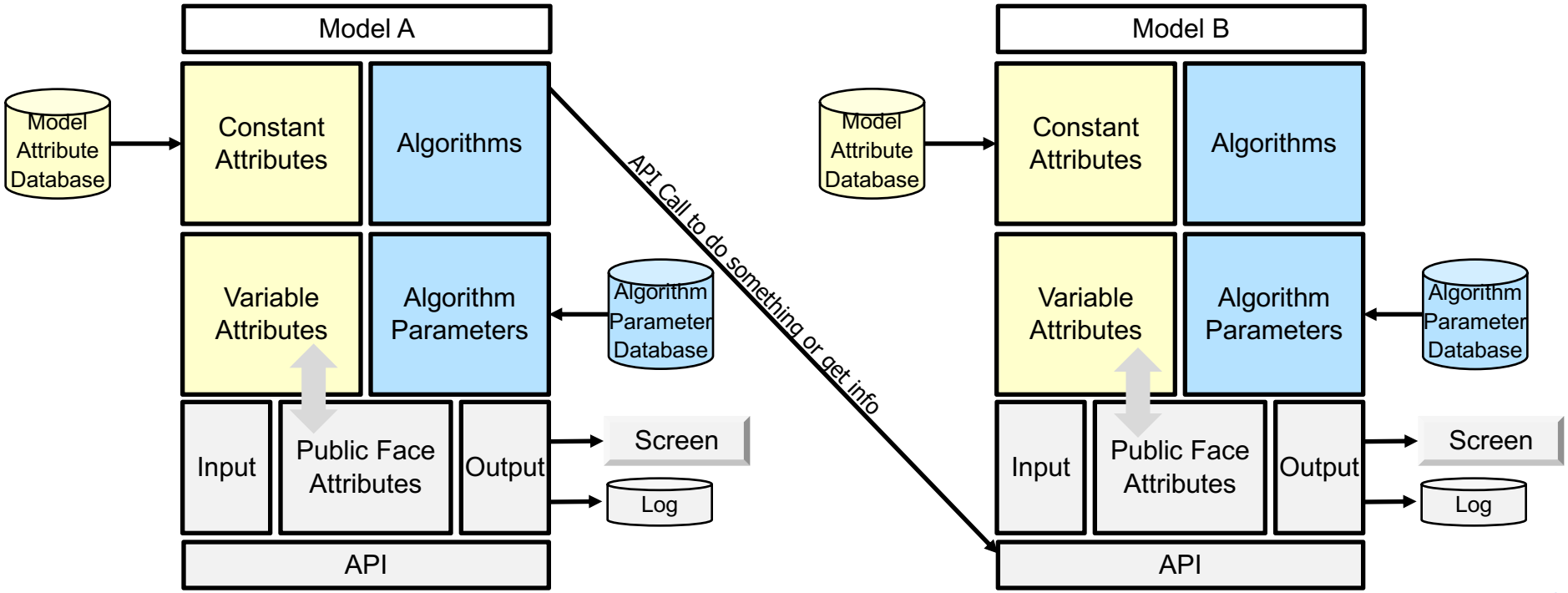
Abstraction
Simplification

Model



Models Interacting

- **When Models interact with one another, they generally don't allow direct access to their attributes by other models**
  - They create a "public face", consisting of that subset of their attributes they allow other models to read
  - They have basic input and output functions, usually including logging
  - They have an API to send and receive messages



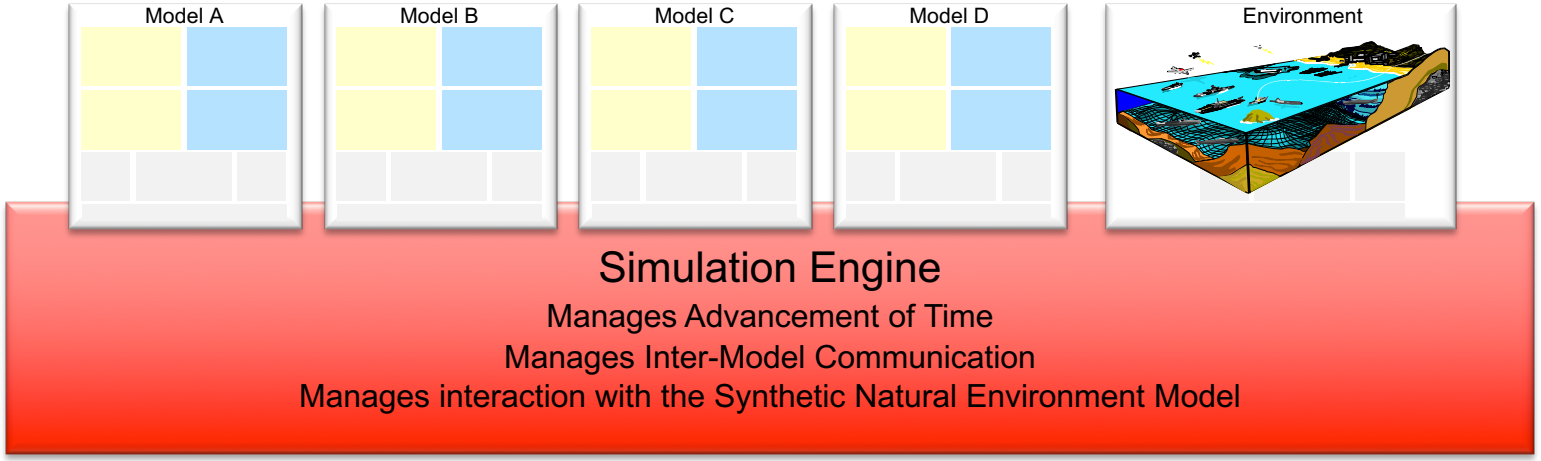
Types of Models in Military Simulation

- **Sensor Models (Optical, Radar, IR, etc.)**
- **Vehicle Movement Models (over land, air, sea, under sea)**
- **Weapons Flyout Models**
- **Weapons Effects Models**
- **Command and Control Models**
- **Human Behavior Models**
- **OODA-Loop Models (observe, orient, decide, act)**
- **Perceived Truth Models**
- **Engineering-level System and Subsystem Models**

- **Model of the Environment (land, sea, undersea, air, space)**
  - Called "Synthetic Natural Environment"
  - Critical for doing *intervisibility* calculations (are two entities within line-of-sight?) and *detection* calculations (did one entity detect another?)

Constructing a Simulation

- **Creating a simulation requires the addition of:**
  - Simulation Engine to manage the advancement of time and inter-model communication
  - The addition of a model of the Environment

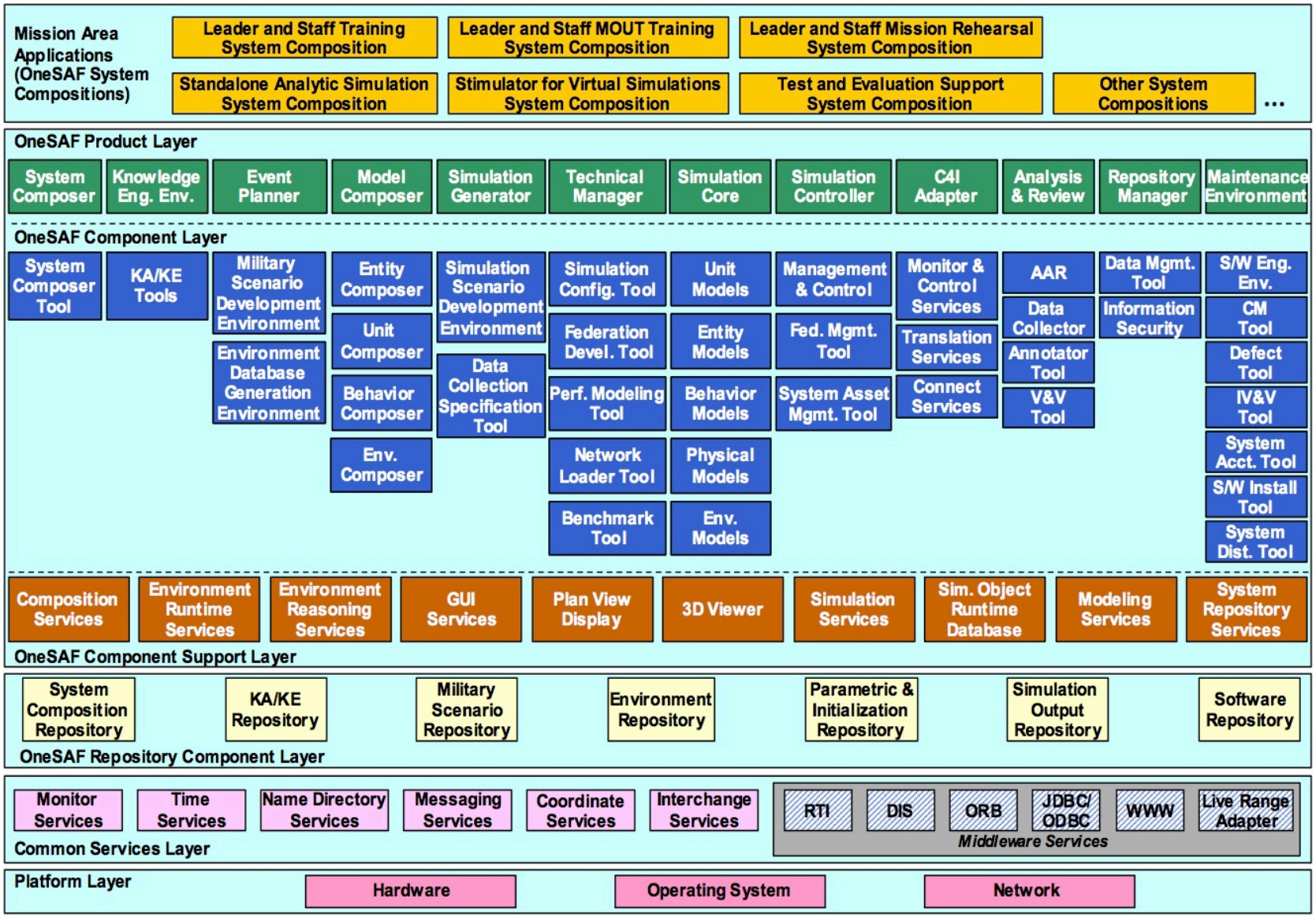


- **The Environment is special – almost all models rely on the environment model**

Simulation Engines

- **There are many simulation engines available for use**
  - Many different features beyond the required ones
  - Many different mechanisms for advancing time
  - Many add a "modeling framework" that constrains developers to develop models in a certain way
- **Examples**
  - OneSAF
  - Flames
  - SPEEDES/WARP IV
  - Open Game World
  - Unreal
  - Delta3D
  - Many, many others

Simulation Systems Can Provide Many Services Beyond the Basics, e.g., OneSAF



Types of Military Simulations

- ***Aggregate*** **simulations model military units as a single aggregate thing, which only interacts with other aggregate things**
- ***Entity-level*** **simulations model military systems at the vehicle and munition level (usually), entities interacting with each other.**
- ***Engineering-level*** **simulations model the individual components of a vehicle/munition system separately, usually to perform engineering analysis**

Distributed Simulations

- **A distributed simulation is one in which multiple simulations are run on multiple computers connected by a network.**

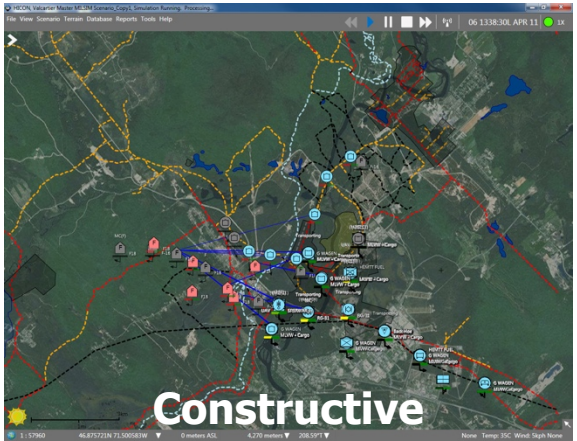
- ***Homogeneous*** **distributed simulation – the same simulation is run on all computers (e.g., OneSAF)**

- ***Heterogeneous*** **simulation – different simulations, usually developed by different development agencies or contractors, and usually containing different assumptions, purposes, and different simulation engines, are run on separate computers.**
  - A set of heterogeneous simulations is called a "***federation***" of simulations

Types of Distributed Simulation

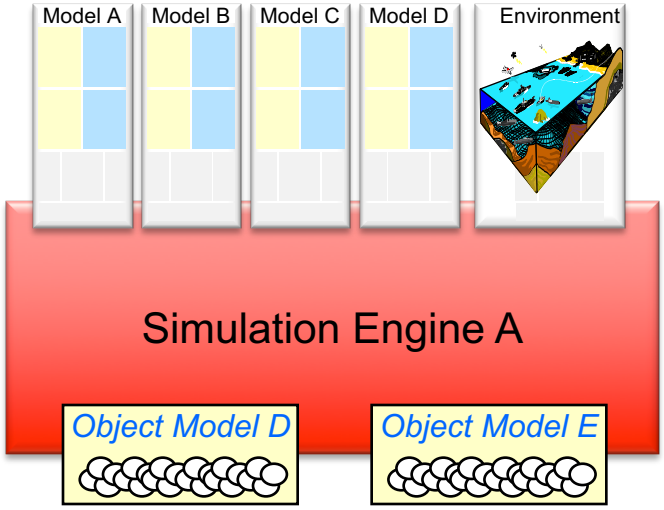
- **"Live" – uses operational personnel and hardware (real aircraft, real ships, real tanks)**
- **"Virtual" – uses operational personnel with equipment that is not operational but preserves operational user interfaces (flight simulator, tank simulator)**
- **"Constructive" – pure computer programs, either controlled by human operators ("semi-automated forces") or run entirely without human intervention ("closed")**

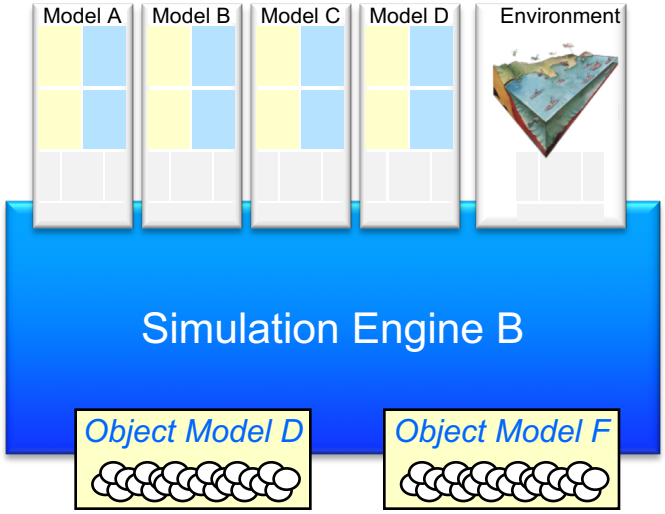
- **There is nothing magical about these classifications, and some simulations may fall into zero or two of these categories**



Public Data Structures and APIs for Distributed Simulation

- **Just like models, entire simulations have a "public face": the set of data structures and APIs about their models that they make available to the rest of the world**
  - We call these public data structures and APIs "***object models***" for historical reasons
  - All of them together represent the "***Federation Object Model***" for that federation

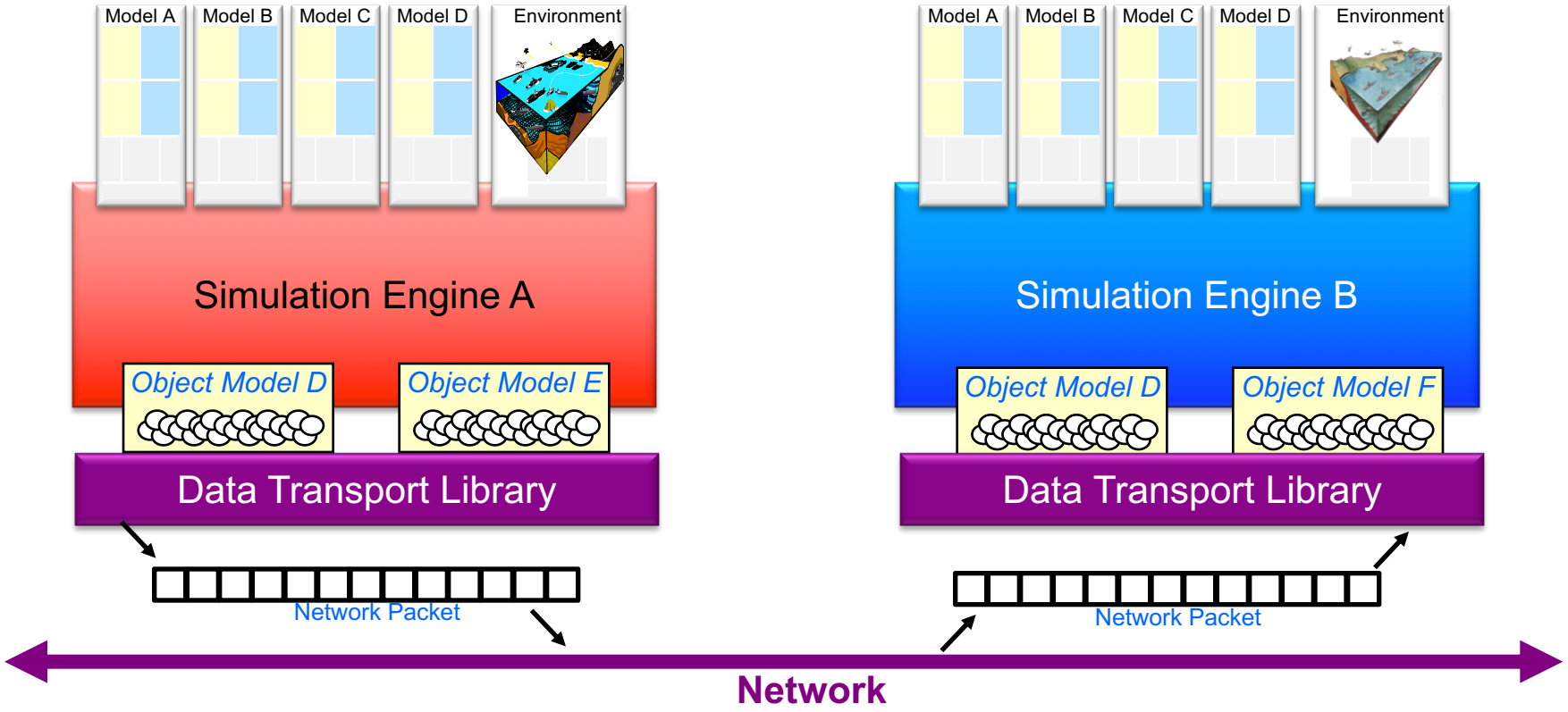




- **Simulations can only interoperate to the extent that they share a common set of public data structures and APIs (object models)**

Common Communication Mechanisms for Distributed Simulations

- **Simulations can only communicate if:**
  - They have a common language to talk with (common object models)
  - They share the same mechanism for encoding this information for transport (data transport software library, also called *Middleware*)
  - The use the same network protocols.

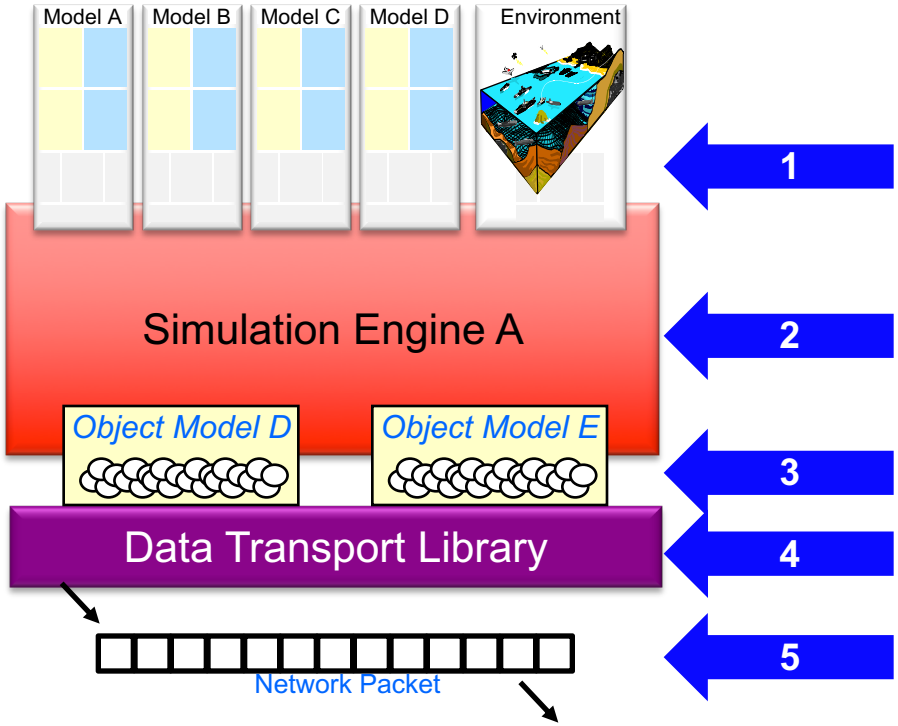


Integrating Independently Developed Simulations

- **Generally these simulations are not built with a knowledge of each other**
- **Built by different contractors with different fidelities for different purposes**
- **Not akin to anything in the commercial world**
- **Different environment models**

It can't be stressed enough that simulations with different models of the environment generally do not play fairly together. Great care must be taken to ensure that their environment models are similar enough to achieve the event's objectives.

Standardization Options



1. **Standardize at the model "public face" and have everything common below (e.g., OneSAF)**
2. **Standardize at the Simulation engine and below (e.g., SPEEDES)**
3. **Standardize at the public data structures and APIs (object models) and below (e.g., HLA and TENA)**
4. **Standardize at the Data Transport Library and below (e.g., VRLink)**
5. **Standardize at the network packets only (e.g., DIS)**

The "higher up" standardization occurs:

- **The more reuse there is**
- **Integration can be simpler**
- **Interoperability can be greatly improved**

But:

- **Simulation developers are more constrained in what they can do**
- **The initial costs are much higher and the risk of failure is greater (JMASS)**

Live, Virtual, Constructive Integration

- **The ability to integrate simulations from each of the three areas.**
- **Extremely tricky due to the unforgiving nature of live systems (hard real time constraints)**
- **Some questions**
  - What does is mean to integrate multiple independently-created simulations?
  - What architecture(s) should you use to actually do the integration?
  - Does it work?
  - What are the known issues?

LVC Integration Issues

- **Issues can be divided up into categories**
  - Fair Fight
    - Terrain Correlation between different L/V/C sims when the sims interact with each other
    - Integrating live sensor systems with simulated entities (e.g., ACES)
    - Integrating live and simulated C4I (messages, voice, video)
  - Software Connectivity
    - Time synchronization (NTP? GPS? HLA/TENA “Time Management”)
    - Bandwidth management (L/V/C interactions/messages within network limits)
    - Object Model and network transport protocol incompatibility
  - Hardware Connectivity
    - Bridging multiple security domains

Differences Between L, V, and C Simulations in a Few Categories

Item	Live	Virtual	Constructive SAF	Constructive Closed	LVC
Individuals	Real	Real	Real/Sim	Simulated	Both
C4 Systems	Real	Real/Sim	Simulated	Simulated	Both
Sensors	Real	Simulated	Simulated	Simulated	Both
Vehicles	Real	Simulated	Simulated	Simulated	Both
Weapons	Real/Sim	Simulated	Simulated	Simulated	Both
EW	Real/Sim	Simulated	Simulated	Simulated	Both
Cyber	Real/Sim	Real/Sim	Simulated	Simulated	Both
Environment	Real	Simulated	Simulated	Simulated	Both
Time	Real	Real	Real	Logical	Real

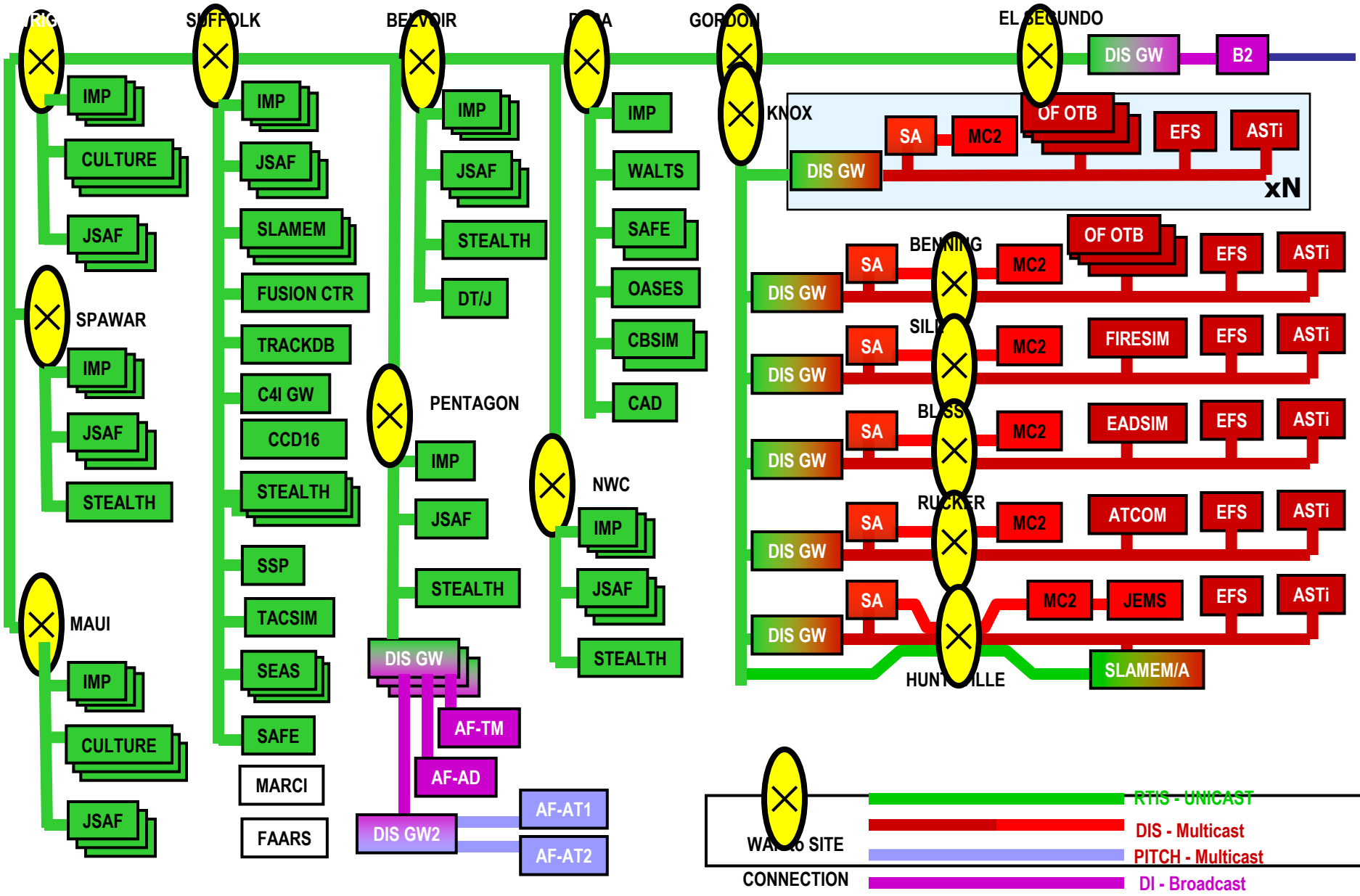
To understand the magnitude of the problem, one needs to understand all of the interactions between all of the items on this chart and manage them so as to produce as fair a fight as possible, for both test and training.


Call of Duty

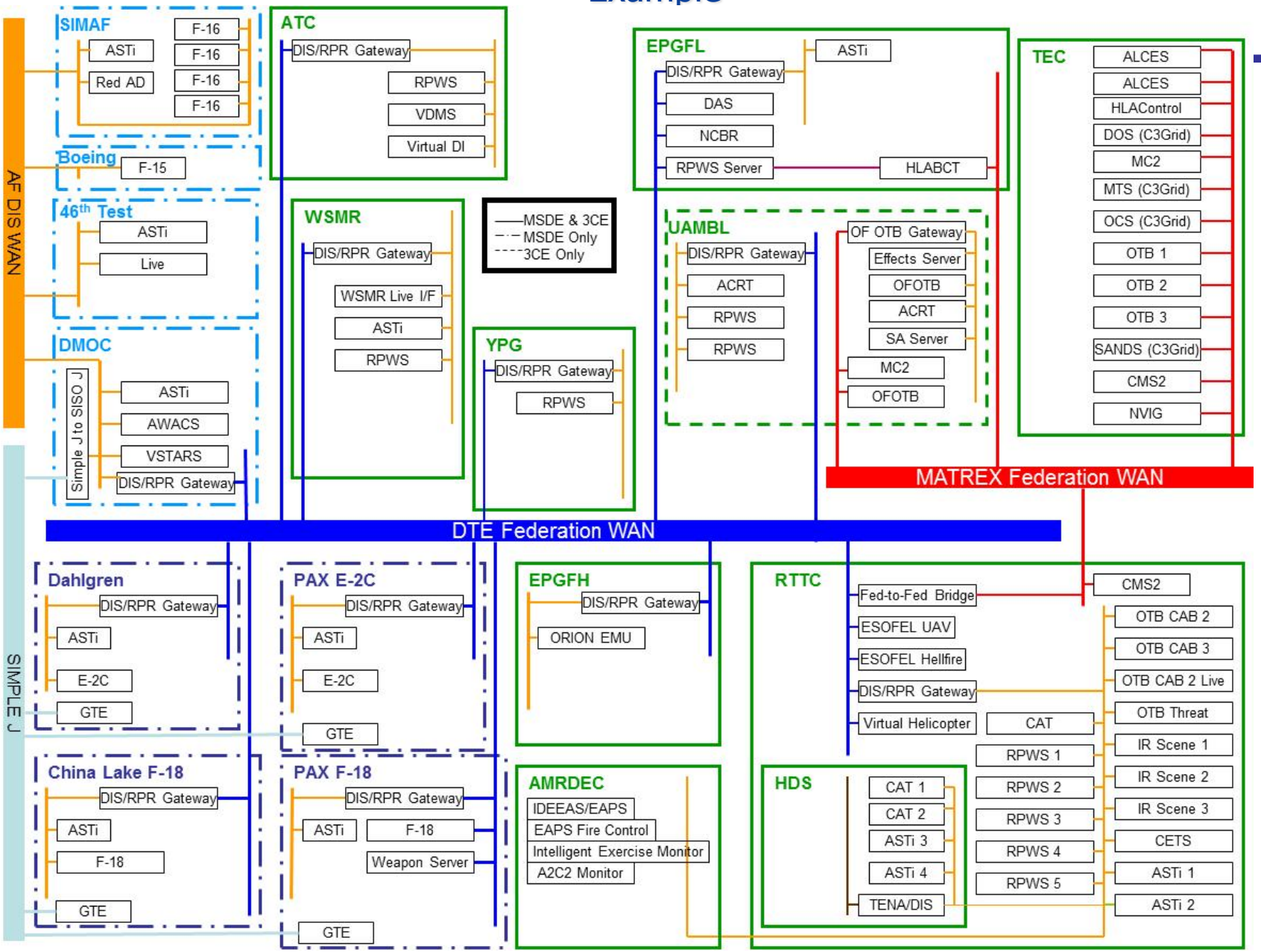

World of Warcraft

What Military Simulation Integrators Do

Integrating Independently Developed Simulations - Urban Resolve 2015 SIM Architecture Example



Integrating Independently Developed Simulations - Distributed Test Event 5 / Multi-Service Distributed Event Example



Four Main Distributed Simulation Integration Strategies

- **Custom integration**
  - Most costly in both the short run and the long run
  - May meet customer requirements better (but probably doesn't)
  - Creates stovepipe
- **Distributed Interactive Simulation (DIS) protocols**
  - Widely used
  - International Standard
  - Large set of "object models" (PDUs) defined
  - Free open-source software libraries available
  - Constrains developers into a single simulation paradigm
- **High Level Architecture for Modeling and Simulation (HLA)**
  - International Standard
  - Contains more functionality and flexibility than DIS
  - Only available commercially
- **Test and Training Enabling Architecture (TENA)**
  - Government-sponsored open standard
  - Contains much more functionality to help developers than HLA or DIS
  - Enforces data contracts using the software language compiler – can't make data contract mistakes
  - All software and support are free to users

Conclusions About Integrating Disparate LVC Simulations

- **Ideally, integrating disparate simulations would entail**
  - One integration architecture
  - One world-view
  - One integration contractors
- **Unfortunately, it's almost always:**
  - Different vendors/contractors
  - Different world-views
  - Different integration strategies/protocols supported
- **Options:**
  - Rewrite some simulations
  - Use gateways from HLA ←→ DIS ←→ TENA ,etc.
    - Lose fidelity
    - Lose measurable fair fight
  - Live with imperfection
- **Accept that multi-architecture events are the norm**
  - Bring in the right systems engineering team

Some Pointers From An Engineer

- **You will never have decent requirements to start from. Consider yourself lucky if you actually get them in writing.**
- **Your requirements will change halfway through your project.**
- **No two individuals will ever agree on what data structures should be used.**
- **If you ever need a common tool, there are at least 15 that will do almost what you need. None of them will do what you need though. You'll have to change what you need or build a 16th tool.**
- **Communication issues are always your software's fault, even when the network is broken.**
- **There is never a reason why network issues get fixed. The network just magically starts working after enough complaints are sent to the network engineers. But there was never anything wrong that anybody can point to.**
- **System administrators will always need a certification to run your software on their machines. Nobody will ever know where this certification is supposed to come from or how one might obtain it.**
- **There's always at least one STIG that would completely prevent any progress in any DoD event. Therefore, general DoD guidelines are to never make progress using computers.**
- **If you ever answer a question about the numerous bizarre protocols you will work with, you're now considered SME on that subject (there aren't that many of us evidently). Be very careful what questions you answer or you might find yourself giving a training in that subject in a few years.**

References

- **Andreas Tolk, ed. et al., *Engineering Principles of Combat Modeling and Distributed Simulation*, Wiley, 2012**
- **Richard M. Fujimoto, *Parallel and Distributed Simulation Systems*, Wiley, 2000**
- **Douglas Schmidt et al., *Pattern Oriented Software Architecture Volume 2: Patterns for Concurrent and Networked Objects*, Wiley, 2000**
- **IEEE Standards for HLA**
  - http://standards.ieee.org/findstds/standard/1516-2010.html
  - http://standards.ieee.org/findstds/standard/1516.1-2010.html
  - http://standards.ieee.org/findstds/standard/1516.2-2010.html
- **IEEE Standards for DIS**
  - http://standards.ieee.org/findstds/standard/1278.1-2012.html
  - http://standards.ieee.org/develop/project/1278.2.html
- **TENA Documentation, Compliance, Software Downloads, Repository**
  - http://www.tena-sda.org/
  - https://www.tena-sda.org/display/intro/TENA+Compliance+Specification
  - http://www.tena-sda.org/repository/

Distributed Interactive Simulation (DIS)

Shamelessly Stolen
and (Slightly) Modified from

Mark McCall, DIS PDG Chair, markmccall@sisostds.org

Don McGregor, NPS, mcgredo@nps.edu

Overview

- **General DIS Overview**
  - DIS History
  - DIS Documents
  - Key Definitions and Concepts
  - PDU Families
- **The Updated DIS Version 7 Standard**
  - IEEE 1278 Update History
  - General standard improvements
  - PDU-specific improvements and new PDUs
  - Annexes

Distributed Interactive Simulation (DIS)

- **Distributed Interactive Simulation (DIS) :**
  - Time and space coherent synthetic representation of world environments
  - Designed for linking the interactive, free-play activities of people in operational exercises
  - Synthetic environment is created through real-time exchange of data units between distributed, computationally autonomous simulation applications
  - Computational simulation entities may be present in one location or may be distributed geographically
- **DIS defines standard Protocol Data Units (PDUs)**
  - Syntax (format) and semantics (rules) for data exchange and simulation interoperability

DIS History

- **August 1989 – First DIS Workshop**
  - Decided to develop DIS using SIMNET as core protocol
- **March 1993 – IEEE Std 1278 approved**
- **Sept 1995 – IEEE Std 1278.1 revision approved**
- **1997 – DIS Workshops replaced by SISO & SIWs**
- **March 1998 – IEEE Std 1278.1a addendum approved**
- **2002 – IEEE 1278.1/1a Reaffirmed**
- **2012 – IEEE 1278.1-2012 Passed (DIS Version 7)**
- **Currently working on DIS 8.**

DIS Documentation Relationships

Distributed Interactive Simulation standards, recommended practices, and related documents

IEEE Std 1278.1-2012
Standard for DIS
—
Application Protocols

IEEE 1278.2-1995
Standard for DIS
—
Communications Services and Protocols

IEEE 1278.3-1996
Rec. Prac. for DIS
—
Exercise Mgt. and Feedback

IEEE 1278.4-1997
Rec. Prac. for DIS
—
Verification, Validation, and Accreditation

SISO-REF-010 Enumerations for Simulation Interoperability

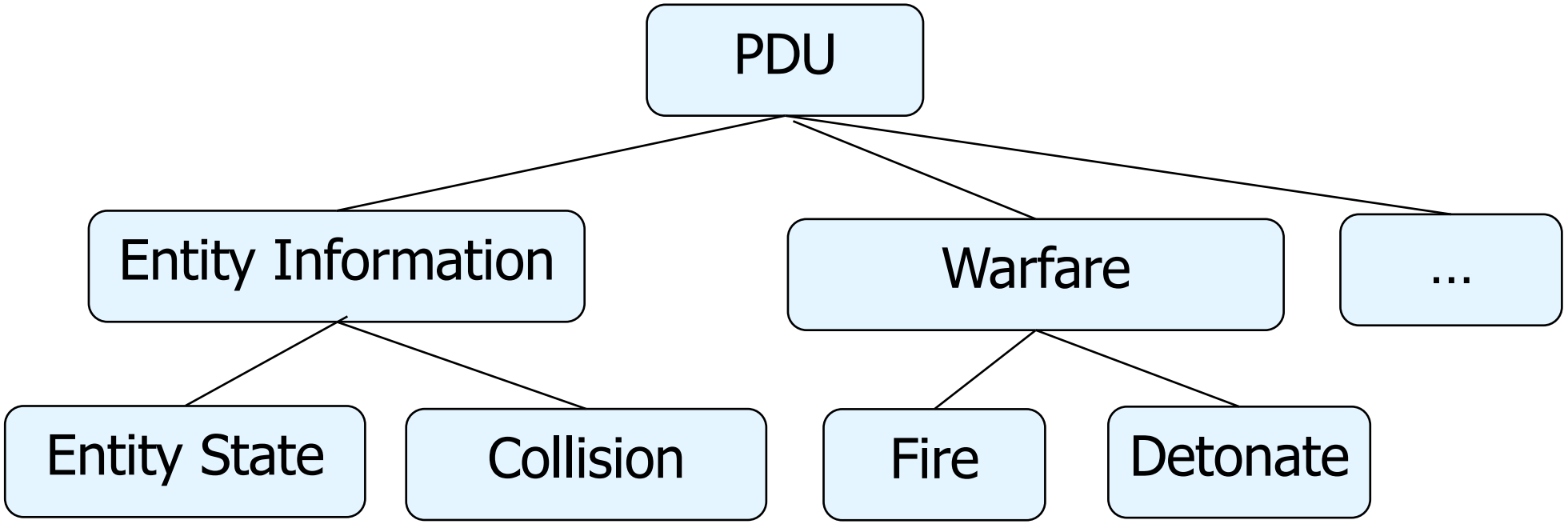
Key DIS Concepts

- No central computer controls the entire simulation exercise
- Autonomous simulation applications are responsible for maintaining the state of one or more simulation entities
- A standard protocol is used for communicating ground truth data
- Changes in the state of an entity are communicated by its controlling simulation application
- Perception of events or other entities is determined by the receiving application
- Dead reckoning algorithms are used to reduce communications processing

Key DIS Definitions

- **Simulation entity:**
  - A physical object in the synthetic environment that is created and controlled by a simulation application and affected by the exchange of DIS PDUs
  - It is possible that a simulation application may be controlling more than one simulation entity
- **Protocol Data Unit (PDU)**
  - A message containing information about the virtual world
  - Sent from one participant to one or more other participants
  - Encoded as a UDP packet on the network in a specific format

DIS Messages



Several dozen different messages (called Protocol Data Units, or PDUs) to describe entity movement, collisions, combat, radio communications, logistics, and more. The Entity State PDU is the most widely used

PDU Families

- Entity information/interaction
- Warfare
- Logistics
- Simulation Management
- Distributed Emission Regeneration
- Radio Communications
- Entity Management
- Minefield
- Synthetic Environment
- Simulation Management with Reliability
- Live Entity
- Non-Real Time protocol
- Information Operations

PDU Families (Cont)

- **Entity information/interaction**
  - Appearance of an entity
  - Location of an entity
  - Entity collisions
  - Attribute PDU (Version 7)
- **Warfare**
  - Weapons
  - Expendables
  - Explosions
  - Fire/Detonate
  - Directed Energy (Version 7)
  - Entity Damage Status (Version 7)

DIS: API

- **DIS doesn't have an API. This seems strange to people coming from HLA or TENA, but reflects common practice in networking protocols**
  - The standardized part is the format of the messages on the wire. The standard is silent about how to create or receive those messages
  - Different DIS vendors have different APIs, but all produce the same format messages. This is in contrast to HLA, which has a standard API, but is silent about the format of messages on the wire. As a result, different HLA RTI vendors usually use different message formats for exchanging information
  - TENA standardizes the API, and there is a single approved implementation of the RTI equivalent; this sidesteps the wire standard problem because there is only one approved RTI equivalent

DIS Example - Send ESPDUs in Java

```
public void sendEspdus()
{
    try
    {
        EntityStatePdu espdu = new EntityStatePdu();

        // Set the entity ID, the unique identifier for an entity in the world
        espdu.getEntityID().setSite(CHINA_LAKE);
        espdu.getEntityID().setApplication(NPS);
        espdu.getEntityID().setSite(1);

        // Set this up as a Russian BMP-1
        EntityType type = espdu.getEntityType();
        type.setEntityKind((short)1);  // vehicle
        type.setDomain((short)1);      // land
        type.setCountry(222);          // Commie russians
        type.setCategory((short)2);    // AFV
        type.setSubcategory((short)1); // BMP
        type.setSpec((short)1);        // basic BMP

        // Set position
        espdu.getEntityLocation().setX(1000.0);
        espdu.getEntityLocation().setY(2000.0);
        espdu.getEntityLocation().setZ(3000.0);

        for(int idx = 0; idx < 100; idx++)
        {
            byte data[] = espdu.marshalWithDisAbsoluteTimestamp();
            DatagramPacket packet = new DatagramPacket(data, data.length, destinationAddress, port );
            socket.send(packet);
            Thread.sleep(1000);
        }

    }
    catch(Exception e)
    {
        System.out.println(e);
    }
```

Entity Coordinates

- **Geocentric Coordinates**
  - Position and Orientation
- **WGS-84 elliptical Earth model**
- **Units in meters and radians**

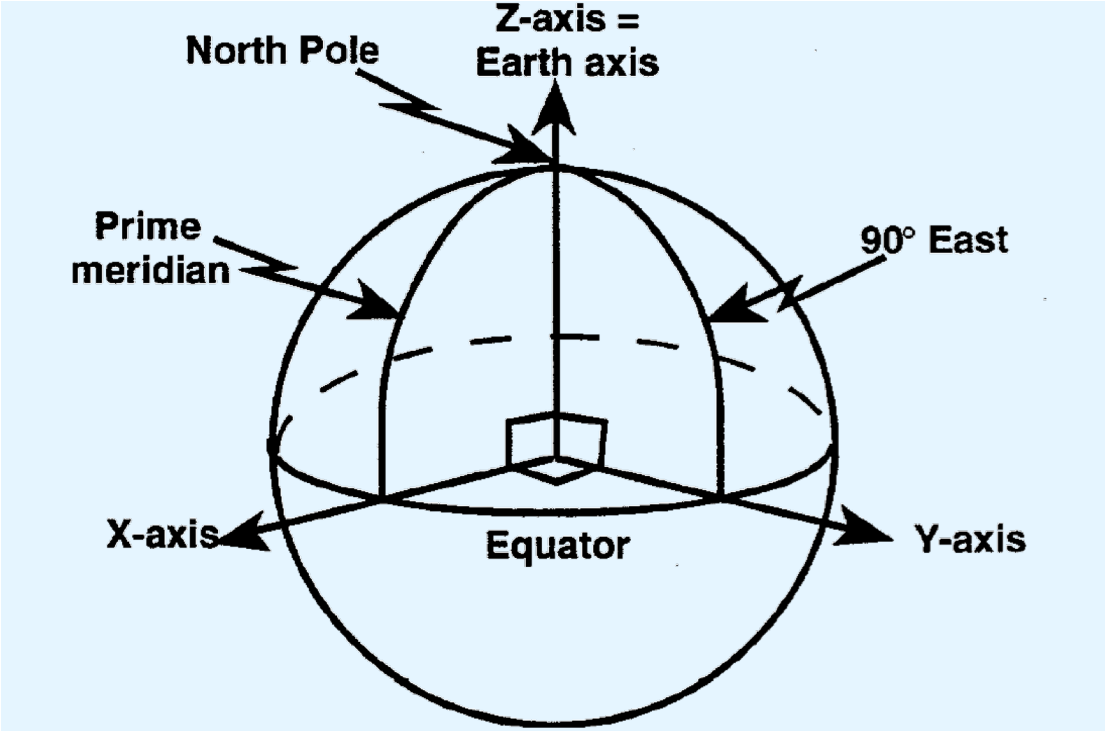


Figure 1—World coordinate system

Entity Type Identification

ENTITY TYPE RECORD	
Entity Kind	8 bit enumeration
Domain	8 bit enumeration
Country	16 bit enumeration
Category	8 bit enumeration
Sub Category	8 bit enumeration
Specific	8 bit enumeration
Extra	8 bit enumeration

- **Hierarchical designation of Entity Type**
- **Enumerations are listed in SISO-REF-010**
  - Over 13,000 entity types

Examples of Type Enumerations

	Kind	Domain	Country	Category	Sub Category	Specific	Extra
F-15C	1	2	225	1	5	3	-
F-15E	1	2	225	1	5	5	-
MiG-27K	1	2	222	2	1	2	-
M1A2 Abrams	1	1	225	1	1	3	-
T-72B	1	1	222	1	2	6	-
D 98 York	1	3	224	4	1	12	-
Mk 44 torpedo	2	7	225	1	9	-	-

Entity Instance Identification

Site	16-bit unsigned
Application	16 bit unsigned
Entity	16 bit unsigned

- **Combination of 3 numbers identify individual entities and objects**
- **Exercises can assign site numbers, sites can assign sims at the site, sims can assign entity numbers**

Example: Entity State PDU (part 1)

Table 131—Entity State PDU

Field size (bits)	Entity State PDU fields	
96	PDU Header	Protocol Version—8-bit enumeration
		Exercise ID—8-bit unsigned integer
		PDU Type—8-bit enumeration
		Protocol Family—8-bit enumeration
		Timestamp—32-bit unsigned integer
		Length—16-bit unsigned integer
		PDU Status—8-bit record of enumerations
		Padding—8 bits unused
48	Entity ID	Site Number—16-bit unsigned integer
		Application Number—16-bit unsigned integer
		Entity Number—16-bit unsigned integer
8	Force ID	8-bit enumeration
8	Number of Variable Parameters (*N*)	8-bit unsigned integer
64	Entity Type	Entity Kind—8-bit enumeration
		Domain—8-bit enumeration
		Country—16-bit enumeration
		Category—8-bit enumeration
		Subcategory—8-bit enumeration
		Specific—8-bit enumeration
		Extra—8-bit enumeration
64	Alternative Entity Type	Entity Kind—8-bit enumeration
		Domain—8-bit enumeration
		Country—16-bit enumeration
		Category—8-bit enumeration
		Subcategory—8-bit enumeration
		Specific—8-bit enumeration
		Extra—8-bit enumeration
96	Entity Linear Velocity	*X*-component—32-bit floating point
		Y-component—32-bit floating point
		Z-component—32-bit floating point

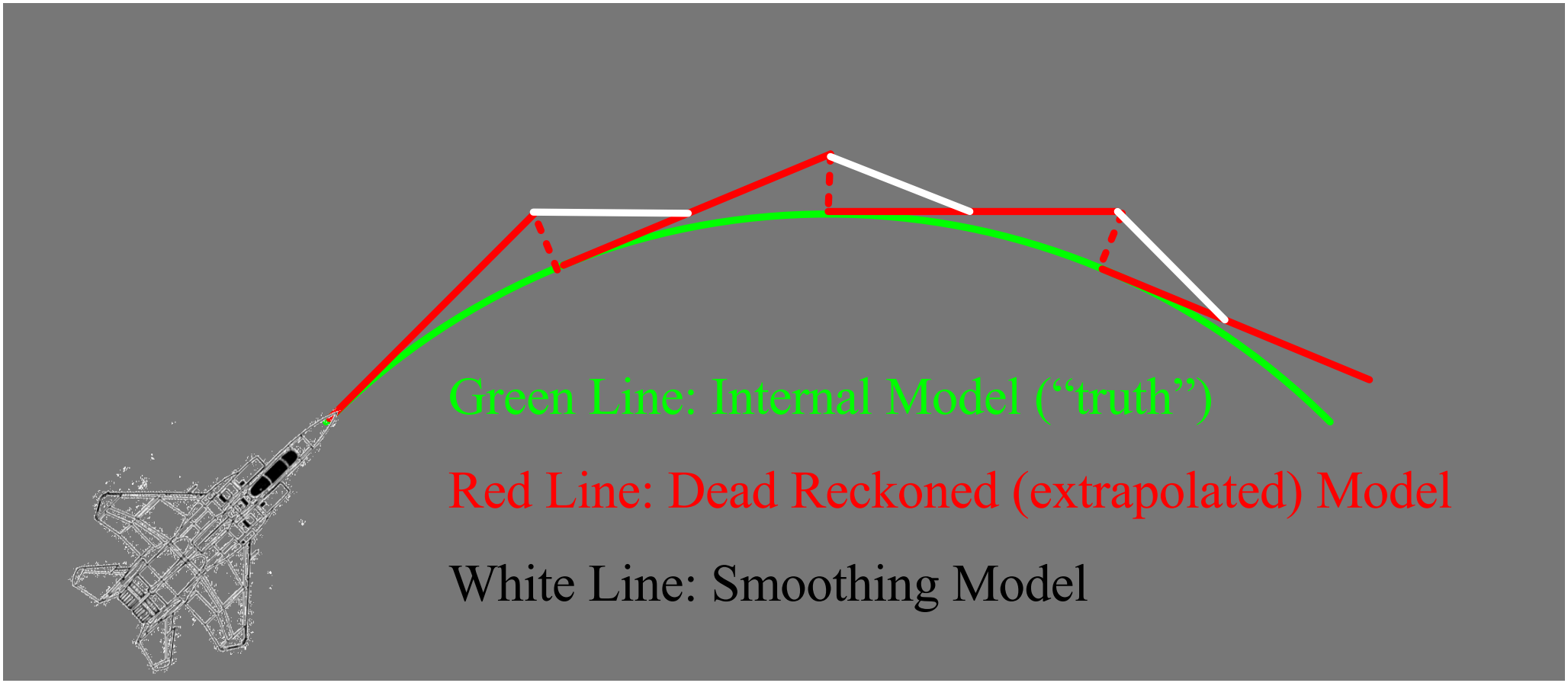
Example: Entity State PDU (part 1

Table 131—Entity State PDU *(continued)*

Field size (bits)	Entity State PDU fields	
192	Entity Location	*X*-component—64-bit floating point
		Y-component—64-bit floating point
		Z-component—64-bit floating point
96	Entity Orientation	Psi (ψ)—32-bit floating point
		Theta (θ)—32-bit floating point
		Phi (ϕ)—32-bit floating point
32	Entity Appearance	32-bit record of enumerations
320	Dead Reckoning Parameters	Dead Reckoning Algorithm—8-bit enumeration
		Other Parameters—120 bits reserved
		Entity Linear Acceleration—3 × 32-bit floating point
		Entity Angular Velocity—3 × 32-bit floating point
96	Entity Marking	Character Set—8-bit enumeration
		11 8-bit unsigned integers
32	Capabilities	32-bit Boolean record
128	Variable Parameters #1	Record Type—8-bit enumeration
		Variable Parameter fields—120-bits
		. . .
128	Variable Parameters #*N*	Record Type—8-bit enumeration
		Variable Parameter fields—120-bits
Total Entity State PDU size = (1152 + 128*N*) bits where *N* is the number of variable parameters		

Dead Reckoning and Smoothing

- Entity sends update when error > threshold
- Receiver extrapolates between updates
- Spatial jump at update is smoothed over



DIS V7 - Extensive clarifications

- **New and improved rules**
  - Lessons learned from 15 years of use
- **New standard is 747 pages**
- **The 1995 and 1998 standards combined were 330 pages**

Even if you have no plans to use any of the new features, the new standard is still extremely useful

Compatibility with Version 5/6

- **Almost every change in the PDU formats and rules are backward compatible with Version 5/6 PDUs**
- **Most changes are also forward compatible (i.e. Version 5 simulations can still make sense of Version 7 PDUs)**
- **Use of former padding fields**
  - New sims can add info, old sims ignore it

Protocol Extensibility

- **DIS now more easily customized**
- **Corrects a weakness in the original standard**
- **Backward compatibility maintained mostly**
- **Variable Parameter Records**
  - Entity State, Detonation
- **Standard Variable Records**
  - Transmitter, IFF, DE Fire, Entity Damage, IO
- **Attribute PDU**
  - Can extend all other PDUs
  - Or, info that doesn't have a PDU

Variable Parameter Records

- The Articulated/Attached Parts record in the Entity State and Detonation PDUs has been opened up for extension
- First 8 bits denotes record
- Other 120 bits is definable
- Still fixed at 128 bit length
- 3 new records so far
- Several ideas for other appearance records

Entity Separation VP Record	
Parameter Type Designator	8-bit enumeration
Reason for Separation	8-bit enumeration
Pre-Entity Indicator	8-bit enum
Padding	8-bits unused
Parent Entity ID	48-bit enum
Padding	16-bits unused
Station Location	32-bit enum

Five New PDUs

- **Directed Energy Weapons**
  - Directed Energy Fire PDU
  - Entity Damage Status PDU
- **Information Operations**
  - IO Action PDU
  - IO Report PDU
- **Attribute PDU**
  - Adds extensibility to the DIS standard

The Attribute PDU

- **Allows existing PDUs to be extended without breaking forward or backward compatibility**
- **The PDU contains sets of Attribute records**
  - Each set is tied to an entity or object
- **Attribute records are open format Standard Variable records**
- **Not allowed to contain information that already exists in other PDUs**
  - Otherwise, there would be confusion about which PDU to use

DIS: Implementations

- **Where can you get a DIS implementation?**
  - Write your own
  - Buy one. There are several commercial implementations
  - Use an open source version
    - Open-DIS (http://open-dis.sourceforge.net)
      - Java, C++, C#, Objective-C, Javascript
    - KDIS (http://sourceforge.net/projects/kdis/)
      - C++
    - Aquaris (http://sourceforge.net/projects/aquariusdispdu/)
      - C++
    - JDIS (http://sourceforge.net/projects/jdis/)
      - Java

New and Improved (in progress) DIS V8

- **The DIS Product Support Group at SISO is currently evaluating changes for the next DIS Standard (DIS v8).**
- **Important changes to consider**
  - 64-bit structures and alignment
  - Little Endian number transmission
  - New timestamp representation
  - Full implementation of PDU variable part structure from DIS v7
- **Participation is open and input from all is welcomed.**

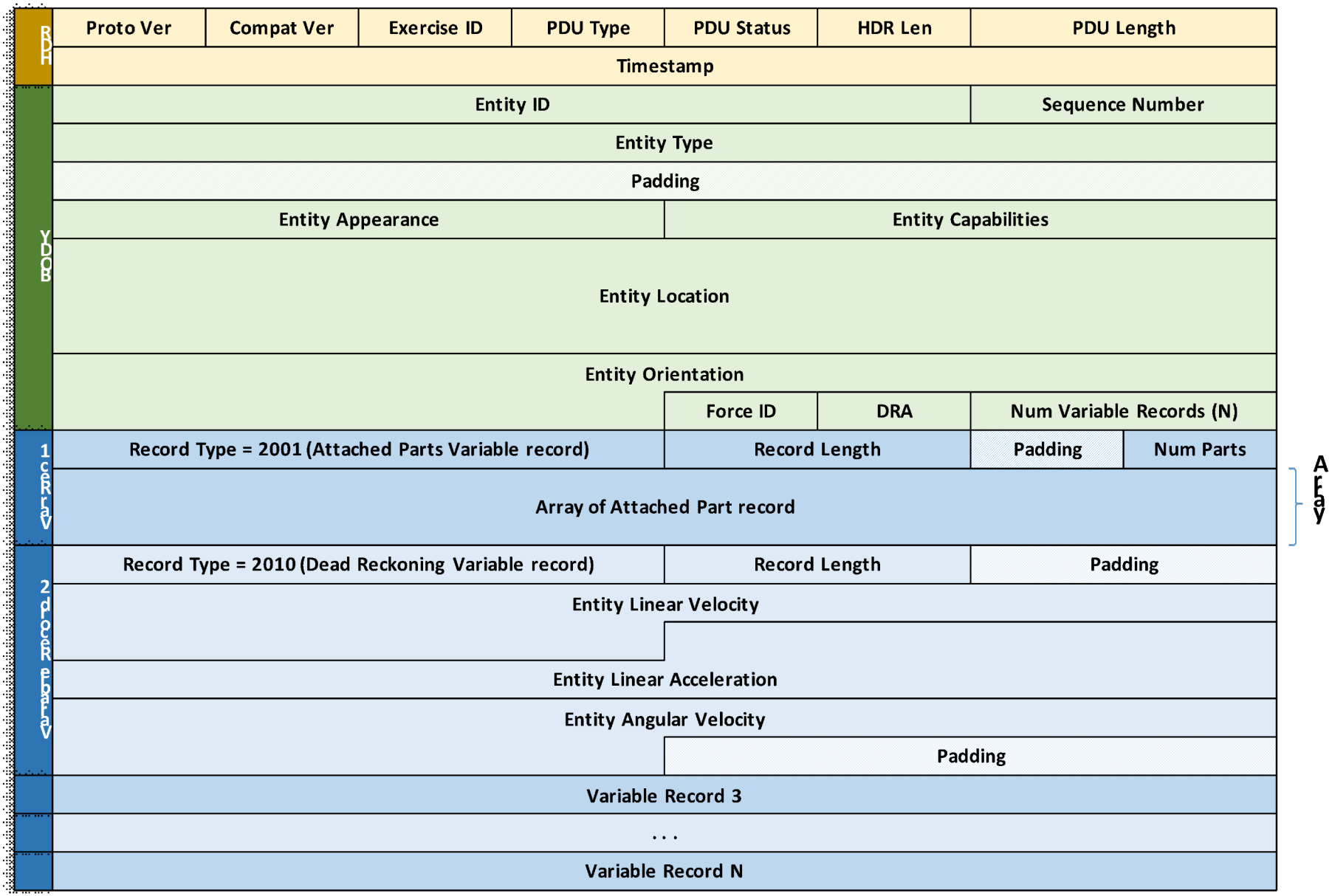
Compatibility Between DIS Versions

- **Backward compatibility: New applications can process old formats**
  - A version number in the protocol provides this
  - The standard does not require this because we cannot enforce requirements on user's software applications
- **Forward compatibility: Old applications can process new formats**
  - Unusual because old applications can't predict the future
  - DIS Version 5, 6, and 7 are mostly forward compatible
  - PDUs were upgraded by adding new info to padding fields
  - But, little padding remains available
- **Gen3 (DIS 8) will break forward compatibility**
  - A clean-sheet design is necessary

Forward Compatibility Beyond Version 8

- **Rules for future PDU changes written in V8**
  - Allows V8 simulators to process V9 and later PDUs
- **New Compatible Protocol Version field**
  - Indicates that a new version PDU is forward compatible
- **PDU header can be expanded in later versions by adding new fields at the end**
  - Rules tell older simulations how to skip over it using a new Header Length field
- **PDU body is partially extensible by adding extra padding in V8**
  - The fixed body *could* use the same length trick as the header, but it doesn't
  - A little extra padding in every PDU is simpler
  - Variable Records can also be used for expansion

DIS 8 Proposed Entity State PDU



Some Other Requested Improvements

- **PCR 276 – Data Representation (Eliminate Endian conversion)**
- **PCR 262 – Simpler voice communications**
- **PCR 217 – Digital Messaging**
- **Remove redundant and unused PDUs**
- **PCR 130 – Redesigned Designator PDU**
- **Support for Higher Fidelity SIMs**
- **PCR 267 – Header**
- **PCR 257 – Improved Timestamp**
  - 64-bit integer includes year, day, hour, etc., with microsecond resolution
- **PCR 258 – Sequence Number to detect dropped PDUs**
- **PCR 260 – Geodetic Coordinates**
- **PCR 261 – Combine Entity/Object to avoid ambiguity**
- **PCR 233 – Machine-readable XML Syntax Language**

Bibliography

- SISO: http://sisostds.org
- SISO DIS Protocol Support Group: http://discussions.sisostds.org/topiclistview.aspx?fid=32
- Open-DIS: http://open-dis.sourceforge.net
- SEDRIS SRM: http://sedris.org
- MaK: http://www.mak.com
- Kdis: http://kdis.sourceforge.net
- Wireshark: http://wireshark.org
- WebGL: http://www.khronos.org/webgl/
- X3D: http://www.web3d.org http://x3dgraphics.com/slidesets
- WebLVC: http://discussions.sisostds.org/topiclistview.aspx?fid=256
- WebSockets: http://tools.ietf.org/html/rfc6455 , http://www.w3.org/TR/websockets

Live-Virtual-Constructive Interoperability Techniques: HLA

Randy Saunders

Agenda

- The High Level Architecture (HLA) Technology
  - Rules of the Road
  - Run-Time Infrastructure (RTI)
  - Object Modeling
- Model Reuse
- HLA Future Growth
  - HLA Evolved Adoption
  - What Comes Next

HLA Technology

- HLA technology is open architecture, free-to-use, defined in open international Standards under IEEE.
  - IEEE Std 1516-2010 Framework and Rules
  - IEEE Std 1516.1-2010 Federate Interface Specification
  - IEEE Std 1516.2-2010 Object Model Template (OMT)
- Commercial suppliers provide software which implements the Interface Specification and tools to facilitate the manipulation of HLA Object Models.
- Migration plans and tools are available to enable transition from legacy HLA 1.3 implementations.

HLA 1516 Framework and Rules

5 Rules for Federations

1. Federations shall have an HLA Federation Object Model (FOM), documented in accordance with the HLA Object Model Template (OMT).
2. In a federation, all simulation-associated object instance representation shall be in the federates, not in the runtime infrastructure (RTI).
3. During a federation execution, all exchange of FOM data among federates shall occur via the RTI.
4. During a federation execution, 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 federate at any given time.

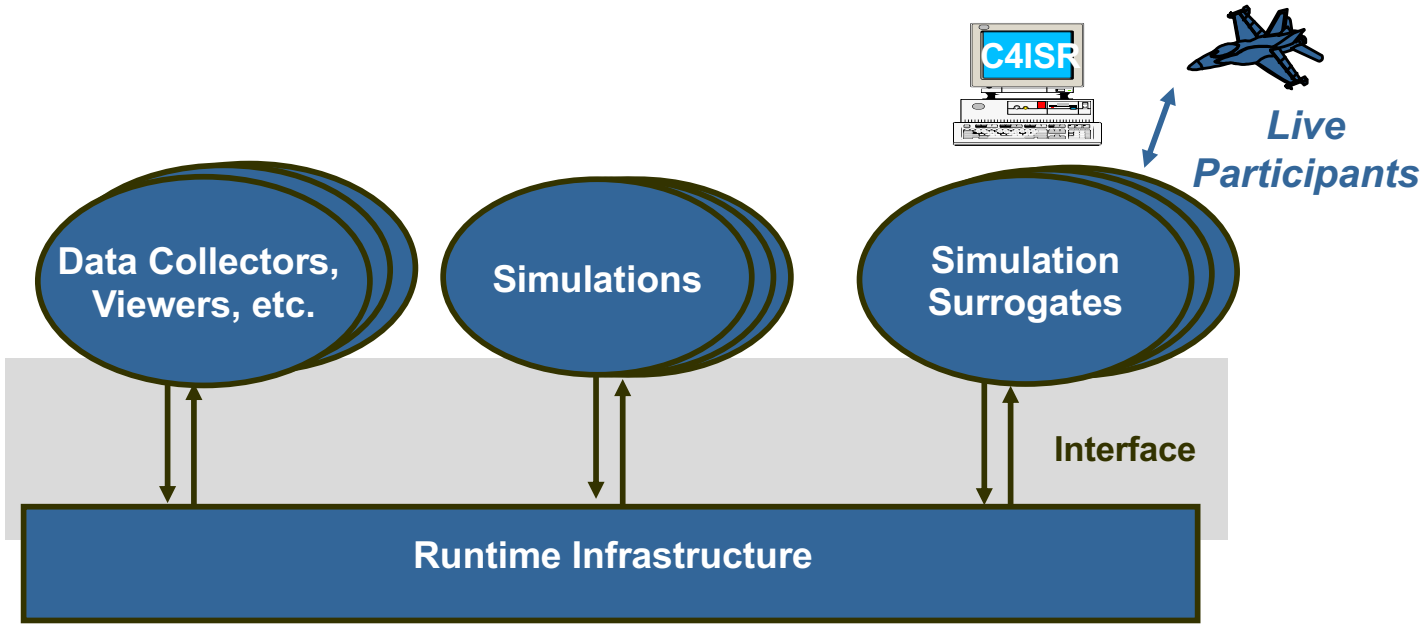
HLA 1516 Framework and Rules

5 Rules for Federates

6. Federates shall have an HLA Simulation Object Model (SOM), documented in accordance with the HLA Object Model Template (OMT).
7. Federates shall be able to update and/or reflect any attributes and send and/or receive interactions, as specified in their SOMs.
8. Federates shall be able to transfer and/or accept ownership of 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 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.

Distributed Architectures

- Existing architectures (DIS, HLA, and others) share the distinction between the Runtime Infrastructure, the Interface, and the simulation

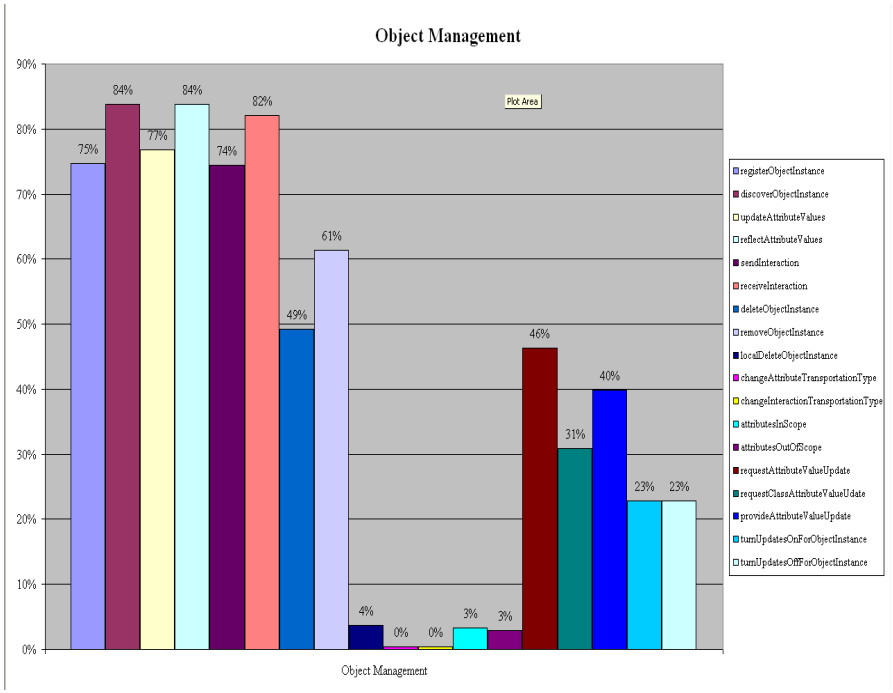


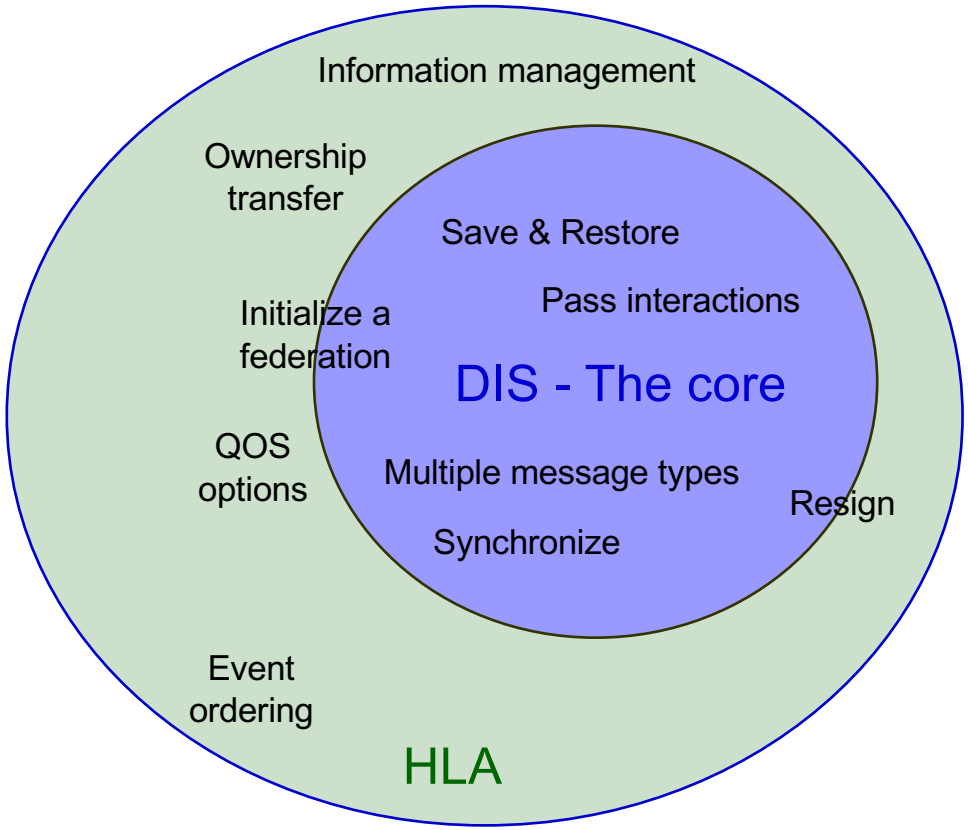
	DIS	HLA
Runtime Infrastructure	Multicast UDP/IP	Commercial RTI
Application Interface	IEEE 1278 Data Units	IEEE 1516 Programmer API

APL

HLA Capabilities

- **Existing architectures have built feature sets that provide for a broad spectrum of capabilities.**
- **Base studies have shown that some features are used much less often than others.**



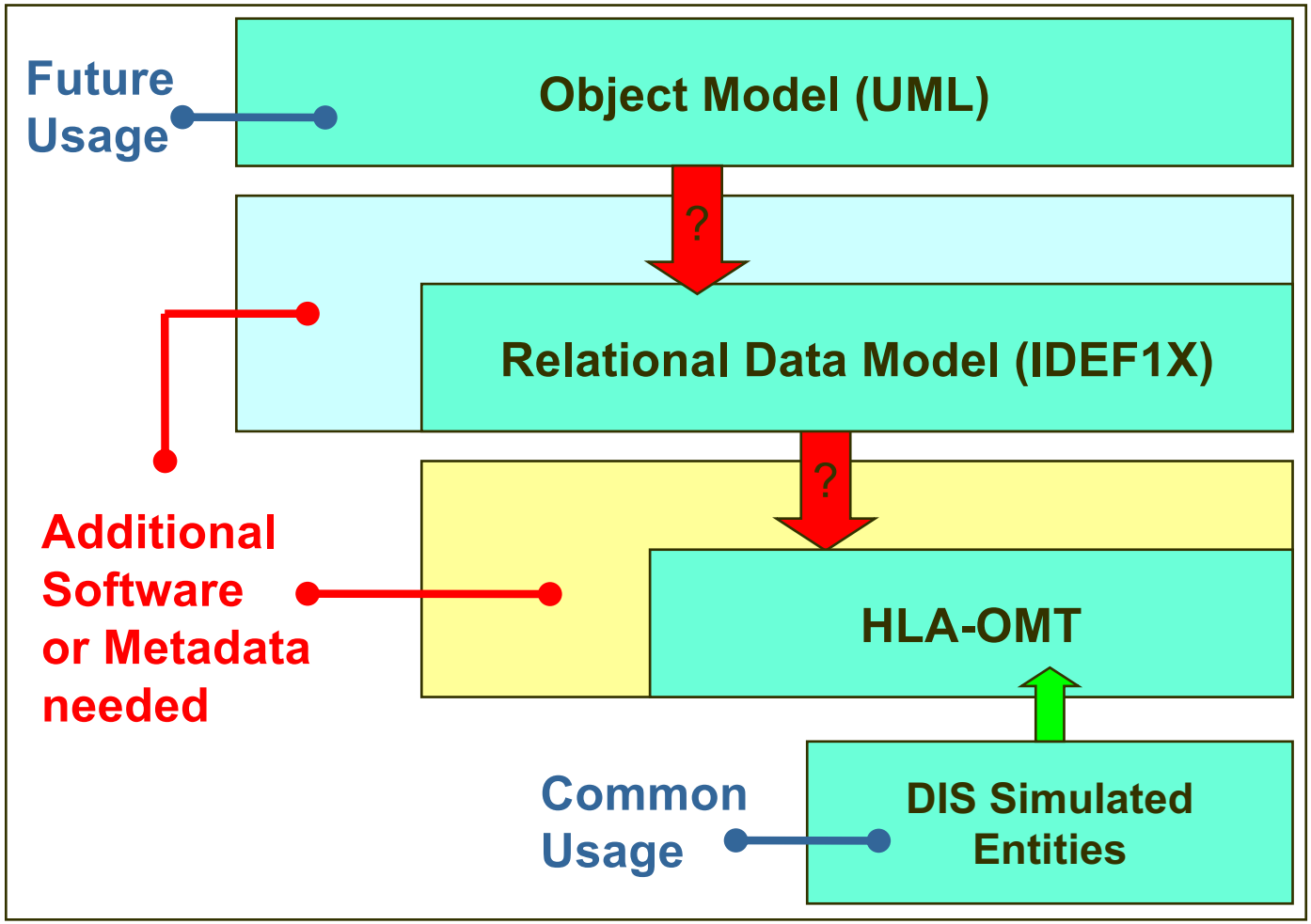


HLA Object Modeling

- An HLA Object Model (FOM or SOM) is not directly relatable to Object-Oriented (OO) software technology.
  - HLA interfaces do NOT require the use of an OO programming language, and they do not exploit OO if used with one.
- HLA Object Models generalize the standardized object model concepts of DIS, and provide structure for optimizing them.
  - DIS had a static data concept, entities had fixed attributes and interactions communicated through a set of Standard messages.
  - HLA provides a mechanism for defining attributes and interactions communicated through Standard API calls.
  - With an OO programming language the objects created have attributes and interactions which are directly used in code statements to form programs.

HLA Object Model Information Content

- HLA object models are a compromise between existing models and future models produced in object-oriented languages.
- OO programs can abstract HLA object model constructs in a few language classes.



Dr. Andreas Tolk 00S-SIW-024

APL

Model Reuse through HLA

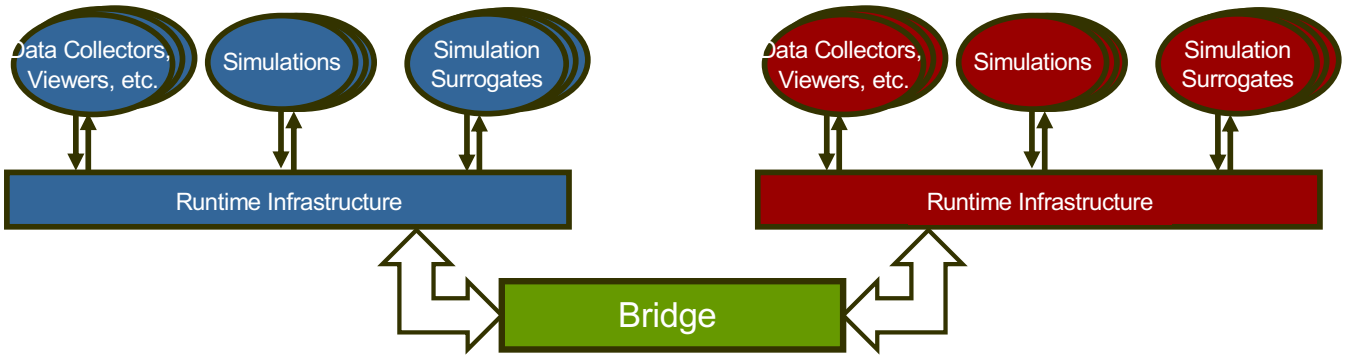
- DSEEP step 3 considers reuse opportunities by examining how existing components can be composed to construct a federation.
- Two composition techniques, commonly referred to as aggregation, are employed in HLA.
- The effectiveness of composition can be limited by monolithic simulations that do not expose component attributes or scale instances efficiently.

Aggregation Types

Paul Gustavson (Simventions)

Reuse Opportunities and Enablers

- The DIS data model represents a set of common features.
  - Standardized representation in HLA through the RPRFOM
  - Standardized representation in TENA through TSPI LROM
- Within an architecture, new models can plug in and establish an initial operating capability.
- Between related architectures a bridge can handle conversions and mapping between simulation representations.

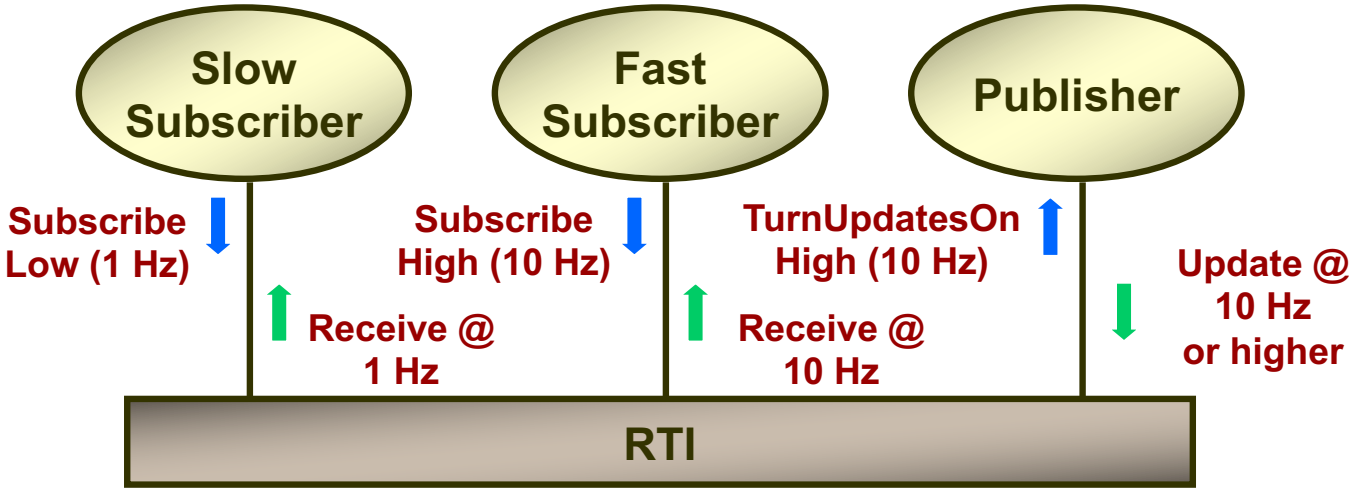


HLA Future

- HLA 1.3 is still in use, like DIS, for connection to legacy federates with this interface.
- HLA 1516-2000 is one active deployment area, commercial production products are available.
- HLA 1516-2010 (HLA-Evolved) incorporated dozens of new features such as fault tolerance, rate control, and modular object models based on user experiences with HLA. This is the HLA you should be using!
- The IEEE P1516 Working Group is currently holding meetings and incorporating comments to revise the Standards for the next version of HLA.
- Future directions are set and funded by HLA users.

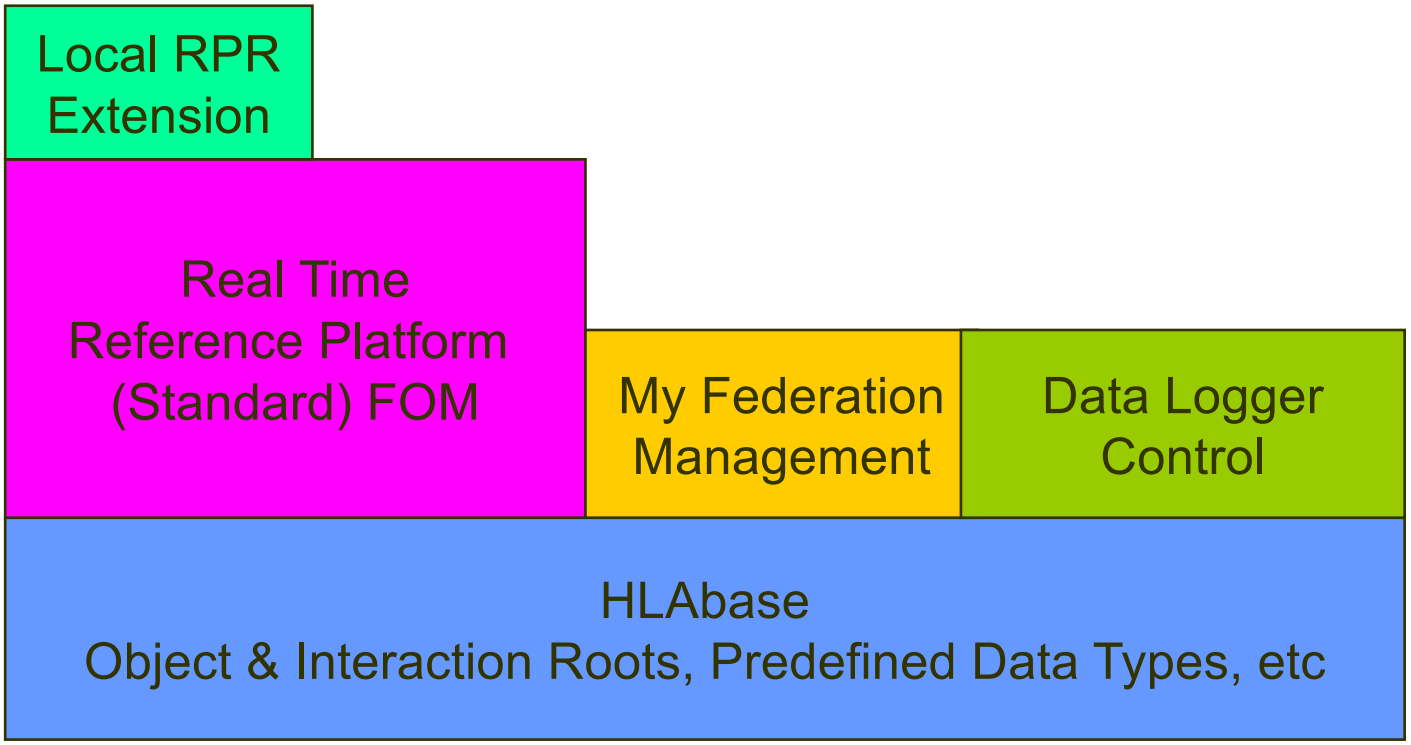
Sample Feature #1 - Update Rate Control

- Federates often have different processing rates, and thus different data needs.
  - Prior HLA versions could burden a low data rate subscriber with discarding updates that were not needed.
  - This results in wasted processing time and network bandwidth along the link to the low data rate subscriber.
  - The RTI handles this situation and sends only needed updates.



Sample change #2 - Modular FOMs

- **HLA FOMs were always modular, but only through copying data from SOMs or other sources into a new FOM.**
  - **The RTI now loads FOM modules incrementally, reducing duplication and FOM distribution logistics.**



New Features being debated

- Clarifications, explanations, and fixes to ambiguities
- Simplified rules for switches and FOM merging
- Relaxation of DDM rules under federate control
- More general datatypes, including variable record structures

- Instance-specific services/interactions
- Fixed structure network protocol between Federate and RTI
- Certificate based Federate authentication

- Join the Working Group and make your needs known.

Practical Considerations

- In designing a distributed solution, cost drivers include:
  - Infrastructure availability
  - Previous integration experience
    - Your developers
    - Your simulations
  - Tools for analysis, testing, and bridging
    - Available for your infrastructure
    - Standards for future evolution
  - Higher levels of conceptual interoperability require better understanding of how the simulations represent the problem
    - Correspondingly higher levels of VV&A effort
  - Intellectual property provisions in the simulations, databases, scenarios, and analysis tools

Future Directions – Up To You

- SISO and the IEEE have re-opened the HLA standards for revision.
- If you join the SISO HLA group you will be able to help identify what ideas are folded into the next version of the standards.
- Current topics include:
  - Service interfaces to HLA objects
  - Object-Oriented programming language objects generated automatically from HLA SOM
  - Defined transport (DDS or something else) compatible with DII COE

Questions

Randy Saunders
Johns Hopkins APL
+1.443.778.3861
<Randy.Saunders@jhuapl.edu>

JMETC and TENA

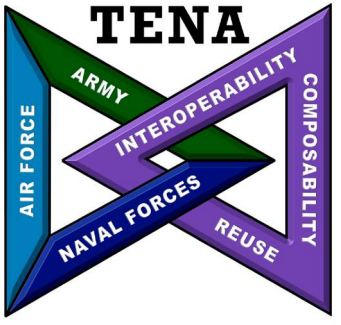
Dr. Edward T. Powell

TENA Architect

What are JMETC and TENA?

- **A corporate approach for linking distributed facilities for test and evaluation**
  - Enables customers to efficiently evaluate their warfighting capabilities in a Joint context
  - Provides compatibility between test and training
- **A core, reusable, and easily reconfigurable infrastructure**
  - Consists of the following products:
    - Persistent connectivity
    - Middleware
    - Standard interface definitions and software algorithms
    - Distributed test support tools
    - Data management solutions
    - Reuse repository

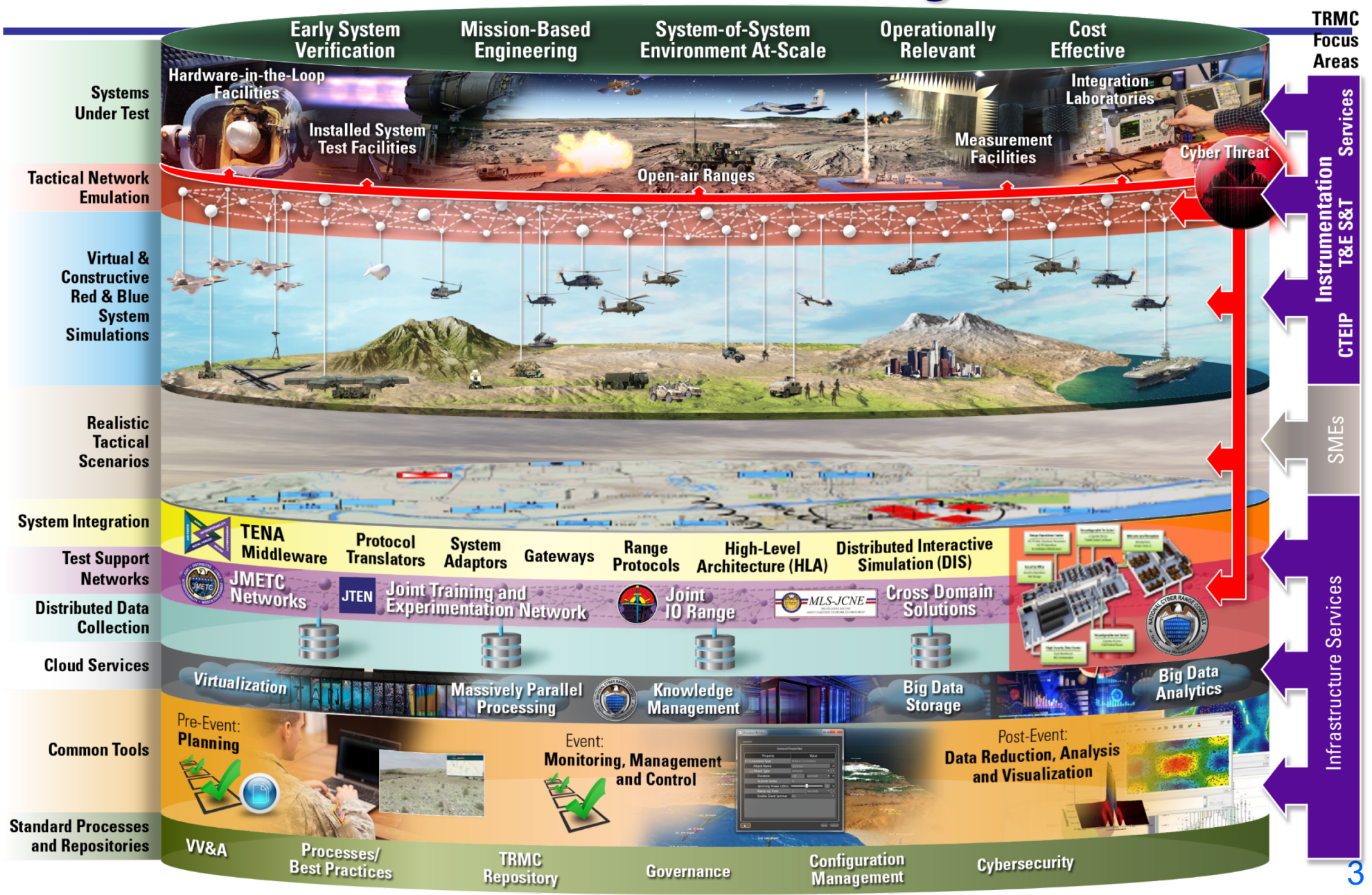
JMETC Networks Using DREN and SDREN



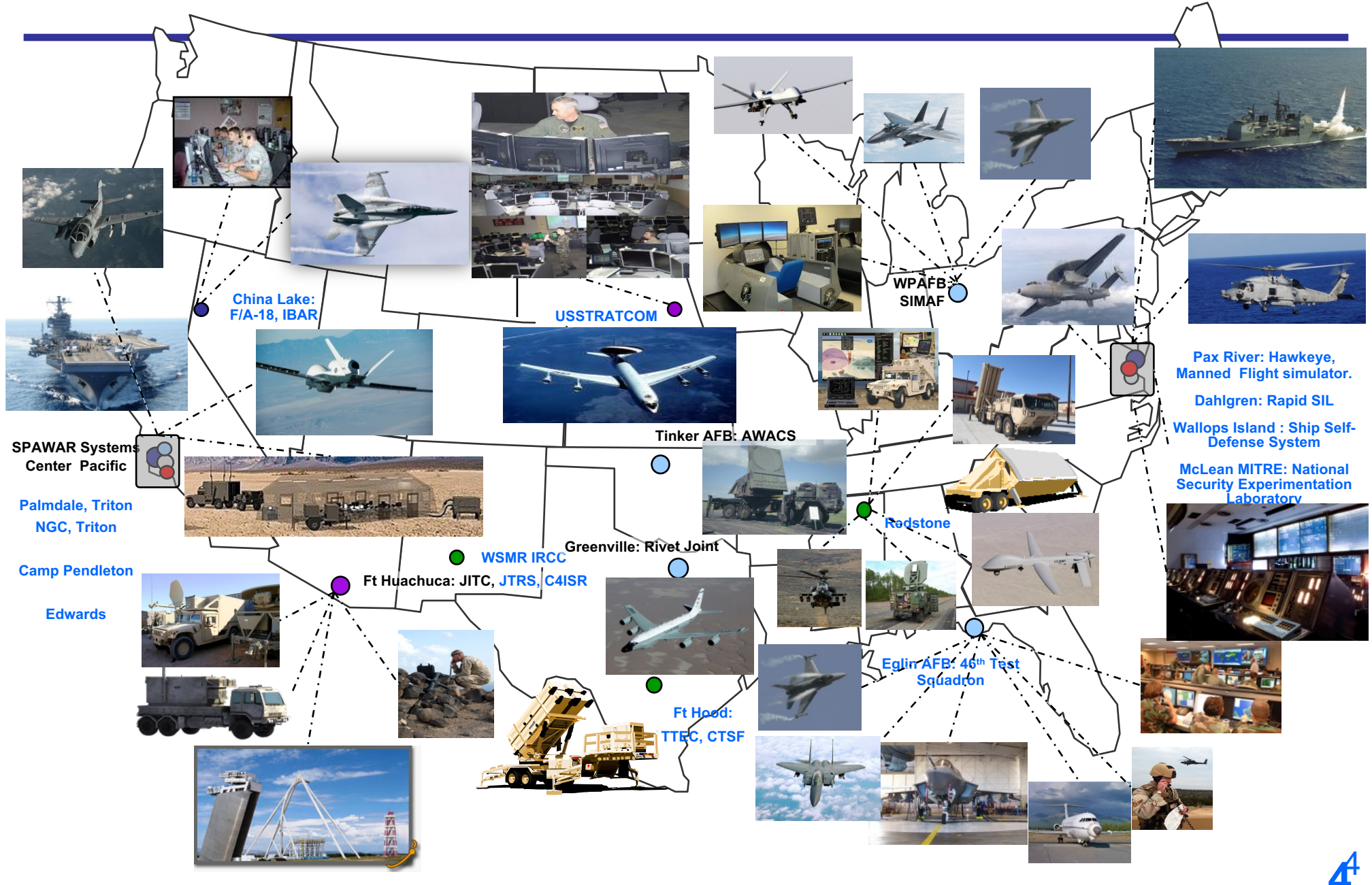
TENA Software, Object Models, Tools, Repository

- **Provides common joint testing process and customer support team for JMETC products and distributed testing**

JMETC Provides an Agile Infrastructure for Distributed Testing



Sampling of Test and Training Assets Available on JSN



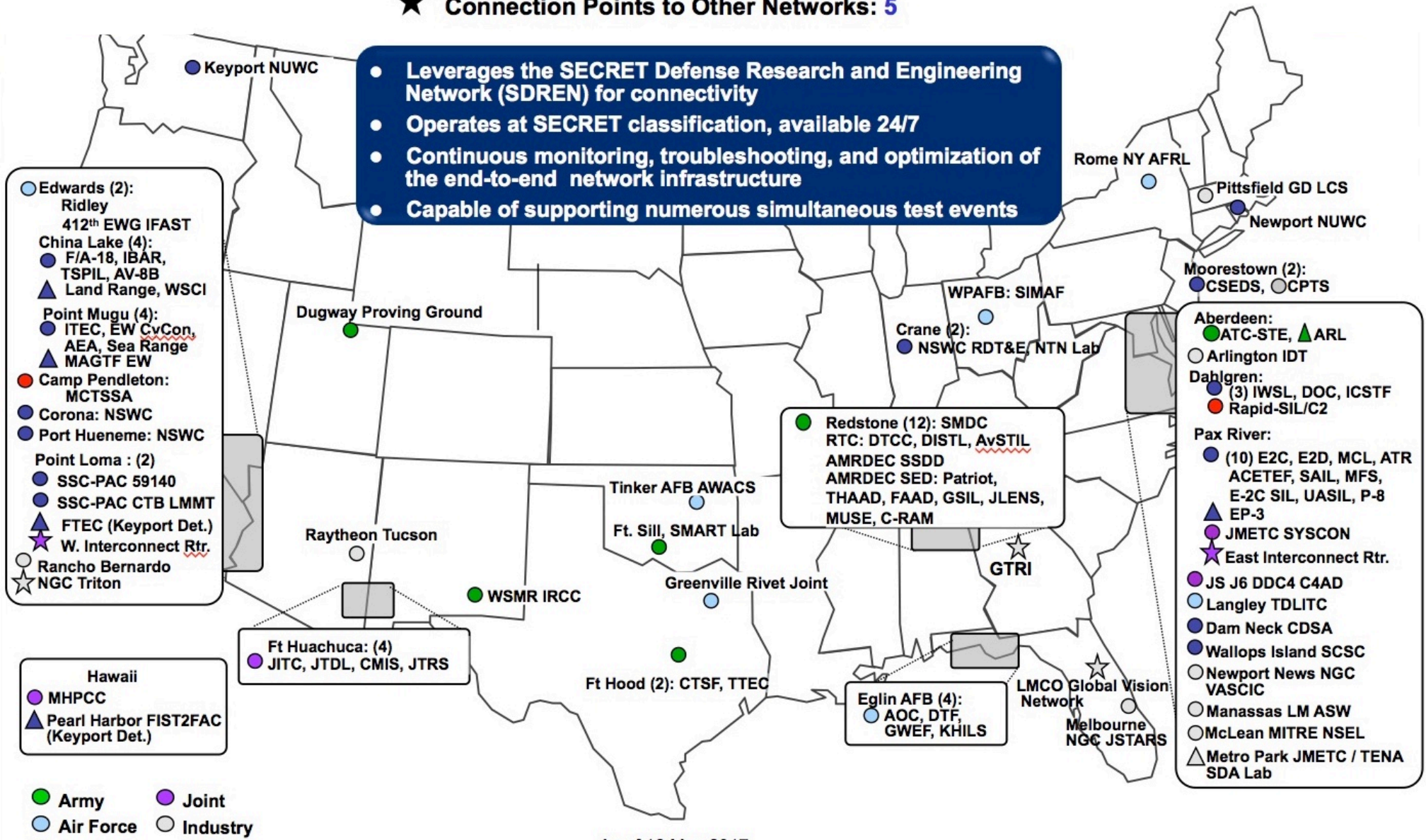
JMETC SECRET Network (JSN)

- **Focus is on persistent connectivity**
  - Standing Agreements
    - All sites have valid Authority to Operate (ATO) and Authority to Connect (ATC)
  - Daily full mesh, end-to-end network characterization ensure optimized performance
  - On demand usage with little to no coordination necessary
    - MOAs in place to authorize connections between all sites
- **Persistency enables user to…**
  - Test capabilities early and often
  - Execute unscheduled/unplanned testing whenever needed
  - Focus on the test rather than the network
- **Operates at SECRET Collateral**
  - Leverages SECRET Defense Research & Engineering Network (SDREN) for connectivity
  - Functional and growing since 2007

Customer time and dollars not spent on infrastructure by leveraging JMETC

JMETC SECRET Network (JSN) Site Map

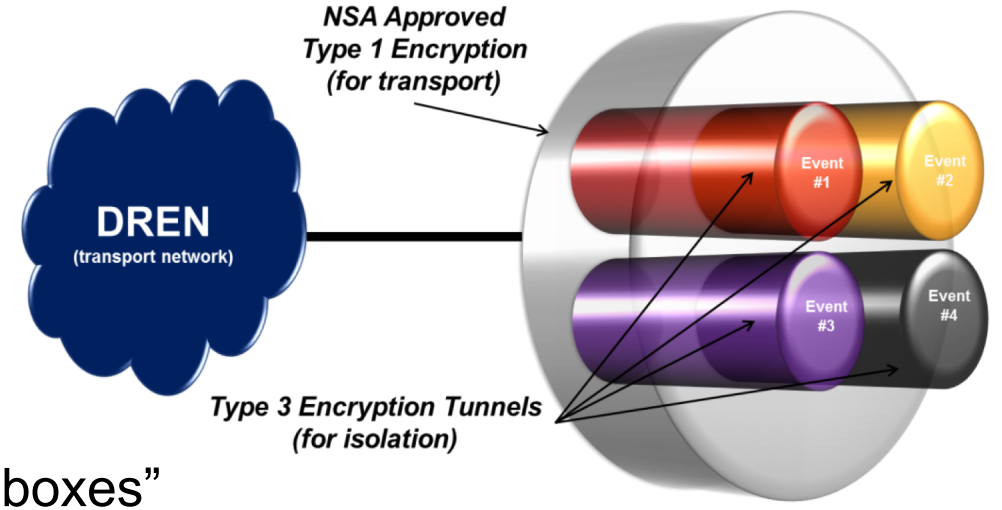
- Functional JSN Locations: 46 (access to 78 labs/facilities)
- Planned JSN Locations: 8
- Connection Points to Other Networks: 5



As of 12 May 2017

JMETC MILS Network (JMN)

- **Focus is on providing secure distributed testbeds to support unconstrained cyber activities and users access to enterprise resources at multiple classifications**
- **Employs Multiple Independent Levels of Security (MILS) architecture**
  - Allows for segregation of data streams by protocol, system, event, COI, etc.
    - Capable of supporting multiple simultaneous events at multiple classifications concurrently
    - Ability to create isolated "sandboxes"



  - Accredited by Defense Intelligence Agency (DIA) to operate from Unclassified up to TS//SCI
    - Included NSA Red Team assessment

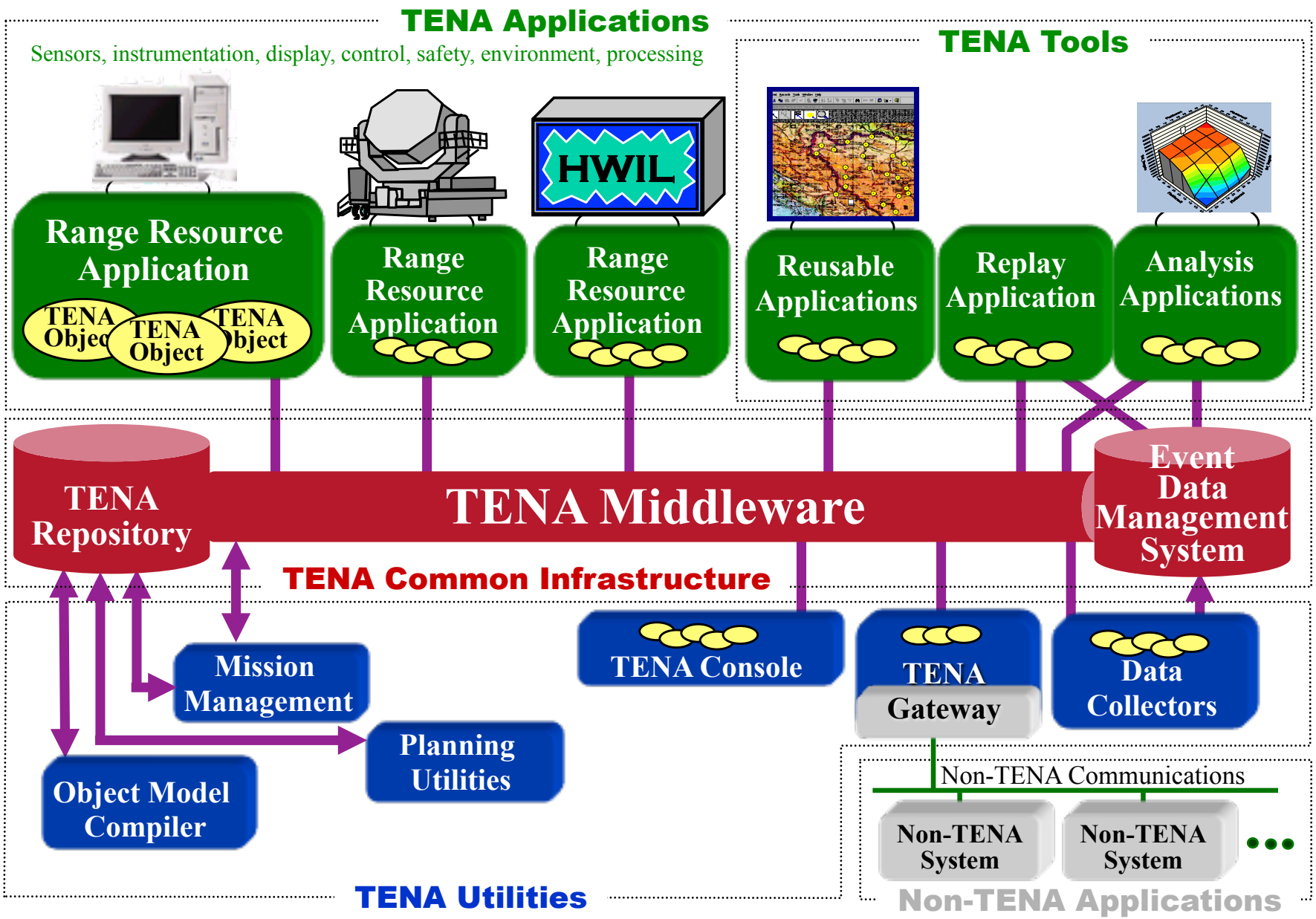
Primarily Used for Cyber Testing

What is TENA?

- **TENA is a software architecture to promote interoperability and reuse for assets in the test and evaluation community.**
- **TENA provides a large number of software libraries and tools to make creating distributed test events easier and more reliable.**

- **Core Architectural Tenets of TENA**
  - Promote Computer Enforceable System Interfaces
  - Utilize Auto-Code Generation to Raise the Abstraction Level
  - Let Computer Detect Interoperability Errors as Early as Possible
  - Design the Middleware to Make it Hard to Use Wrong
  - Anticipate Better Techniques and Technologies
  - Emphasize Live-Virtual-Constructive Interoperability

TENA Architecture Overview



Benefits of TENA

- **All TENA software and support is free to users**
- **Now supports both C++, Java, and .Net!**
- **TENA software is thoroughly tested and very reliable**
- **TENA Auto-Code Generation makes creating a TENA application as simple as possible**
  - TIDE Tool manages installation and configuration, upgrading and maintenance
  - Auto-generated starting points mean you never start with a blank page
  - Rapid development of real-time, distributed, Live-Virtual-Constructive applications
  - Auto-generated test programs make integration a snap
- **TENA's technical approach emphasizes cost savings and reliability**
  - The TENA software is hard to use wrong
  - TENA catches many user errors at compile time rather than run time
  - TENA Tools provide unprecedented understanding of an event
- **TENA has a standard object model enhancing interoperability**
- **The TENA web site/repository has extensive documentation, training, and collaboration capabilities**
- **TENA has a plan for evolution and funding to execute this plan!**

TENA Objects are Compiled In

- **Why use compiled-in object definitions?**
  - **Strong type-checking**
    - Don't wait until runtime to find errors that a compiler could detect
  - **Performance**
    - Interpretation of methods/attributes has significant impact
  - Ability to easily handle complex object relationships
  - Conforms to current best software engineering practices
- **How do you support compiled-in object definitions?**
  - Use a language like CORBA Interface Definition Language to define object interface and object state structure
  - Use **code generation** to implement the required functionality
- **Thus the concept of the TENA Definition Language (TDL) was created**
  - Very similar to IDL and C++

How hard is it to create a new TENA Object Model?

1. Name the object model, including the version
2. Define the message or object types needed by the application
3. Define the attributes that characterize the messages and objects
4. Determine if any attributes are constant or optional
5. Define any remote or local methods

```
  file Example-Vehicle-v6.tdl
package Example {
  enum Team {
    Team_Red,
    Team_Blue,
    Team_Green };

  class Vehicle {
    optional string name;
    const Team team;
    float64 xInMeters;
    float64 yInMeters;
    driveTo (float64 xInMeters,
             float64 yInMeters);
  };
};
```

TENA has a powerful meta-model for defining expressive object models, yet descriptive models are easy to create

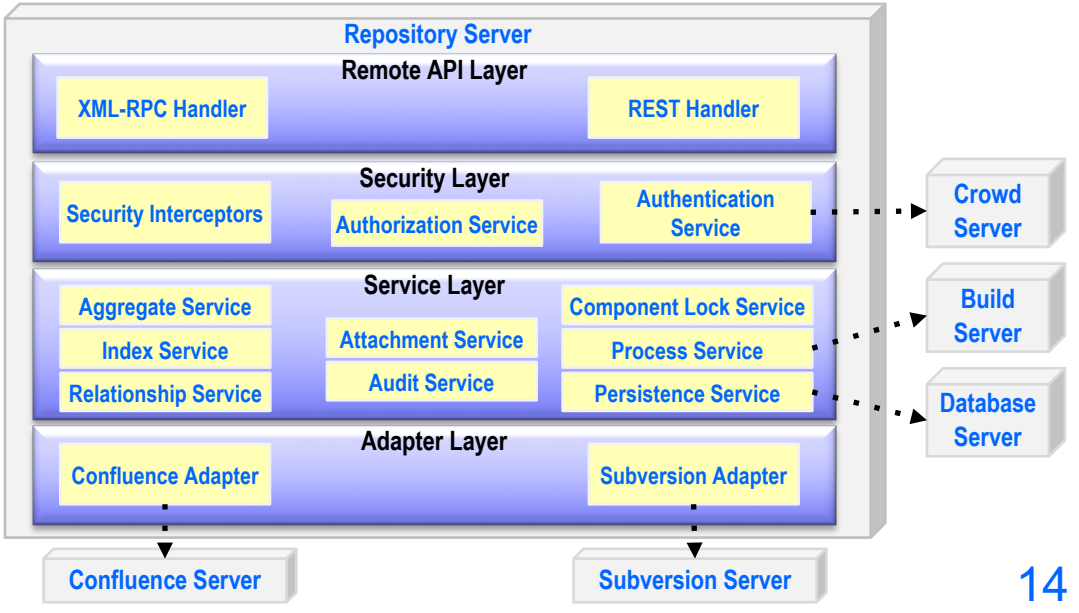
TENA Standard Object Models:
A Common Set of Data Definitions for the Entire Range Community

- **Platform Related**
  - TENA-Platform-v4
  - TENA-PlatformDetails-v4
  - TENA-PlatformType-v2
  - TENA-Embedded-v3
  - TENA-Munition-v3
  - TENA-SyncController-v1
  - TENA-UniqueID-v3
- **Time-Space Position Information (TSPI) Related**
  - TENA-TSPI-v5
  - TENA-Time-v2
  - TENA-SRFserver-v2
  - TENA-Pointing-v1
- **JNTC OMs (for Training)**
  - JNTC-AirRange-v2
  - JNTC-CounterMeasure-v2
  - JNTC-IndirectFire-v2
  - JNTC-Instrumentation-v2
  - JNTC-NBC-v2
  - JNTC-ObstacleMinefield-v2
  - JNTC-Threat-v2
- **Others**
  - TENA-AMO-v2
  - TENA-Engagement-v4
  - TENA-Exercise-v1
  - TENA-GPS-v3
  - TENA-Radar-v3
- **In Progress**
  - Range Instrumentation OM Suite
  - TENA-AVstream
  - TENA-LiftoffDetector
  - TENA-Link16
  - TENA-PowerController
  - TENA-SpectrumAnalyzer
  - TENA-Telemetry
  - TENA-Waypoint
  - TENA-Weather
  - TENA-LVC-Emitter
  - Additional JNTC OMs for training

TENA Repository

- **Purpose: to contain all reusable TENA data**

- **Current Repository Contents:**
  - All TENA Object Models, both standard and user-designed
  - All TENA software (middleware, helpdesk cases, tools, gateways, reusable applications, and reusable components)
  - All TENA documentation
  - Provide an easy-to-use secure interface to all of this information
- **The Repository is a collection of services and technologies based around a wiki-like front end using REST and XML-RPC**

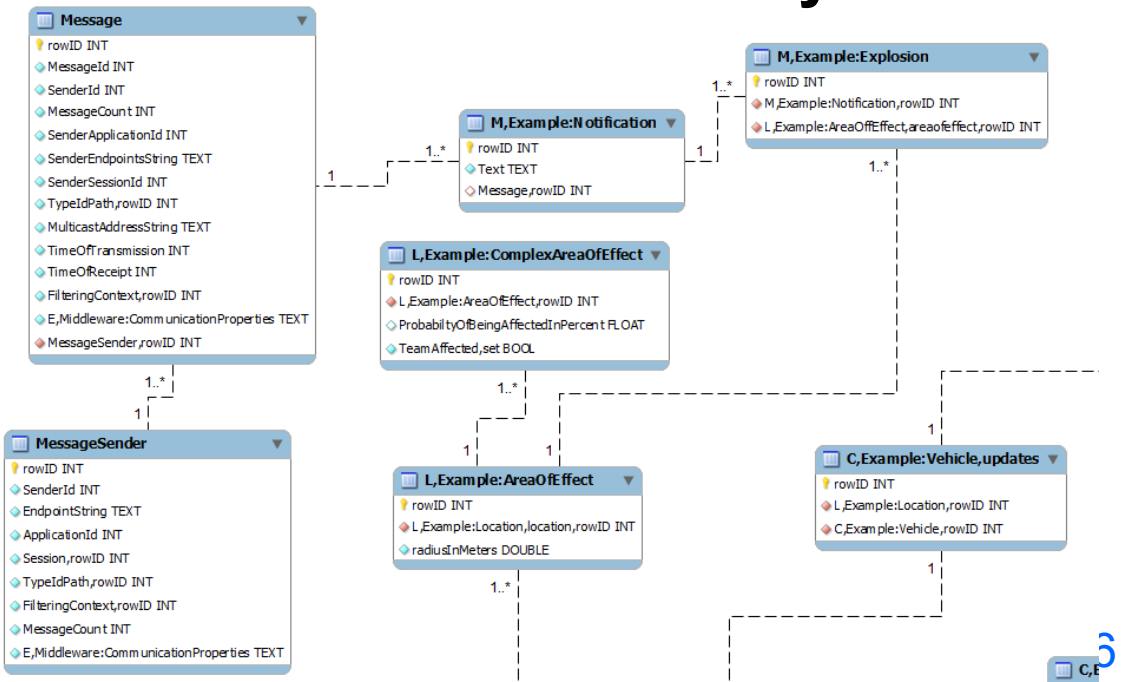


TENA Middleware Purpose and Requirements

- **Purpose: high-performance, real-time, low-latency communication infrastructure used by range resource applications and tools during execution**
- **Requirements:**
  - Fully support TENA Meta-Model
  - Be easy to use
  - Be highly reliable
  - Many varied communication strategies and media
    - Including management of quality-of-service
    - Including object-level security services
  - Be high-performance, including
    - Support multiple information filtering strategies
    - Support user-defined filtering criteria
  - Support a wide variety of range-relevant platforms (HW/OS/compiler)
  - Be technology neutral

TENA Data Collection System Purpose and Requirements

- ***Beta*** **data collection system released**
- **Supports**
  - Collecting arbitrary Object Model information
  - Contains data viewer
  - Contains playback capability
- **Currently works with MySQL and SQLite database systems**
  - Database schema follows the structure of the object model, with separate table for each object and message type
- **Separate Data Collector Application**
  - Export to Excel for viewing
- **Playback application**

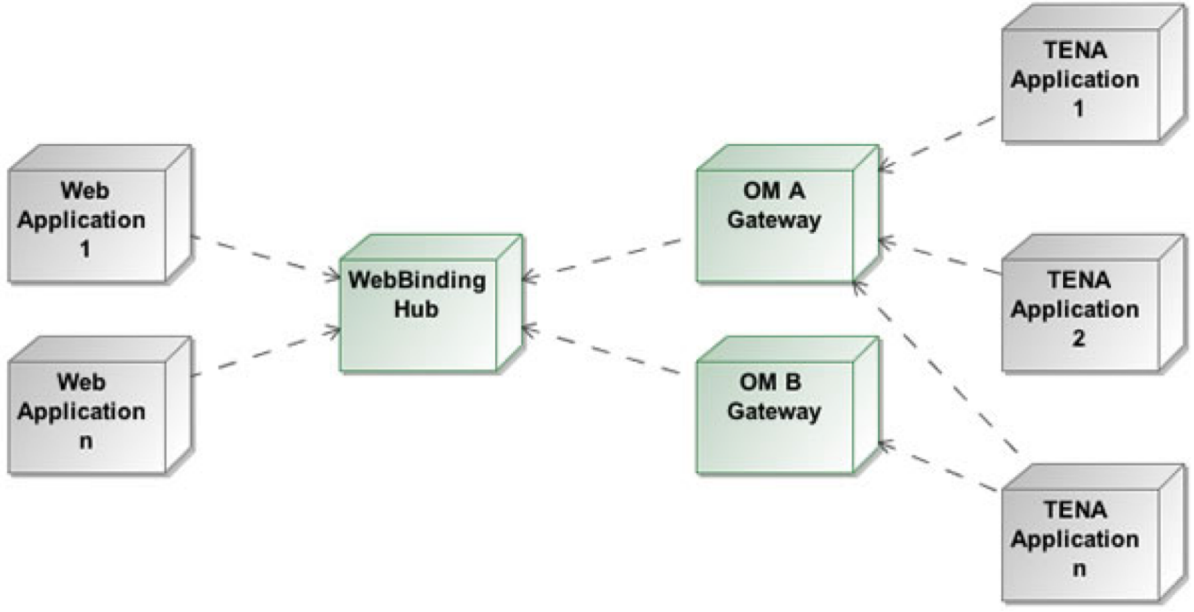


TENA Provides Free GOTS LVC Tools (Partial List)

- **TENA Utilities—Making TENA easier to use**
  - TENA Repository (automated software building, community source code collaboration)
  - TENA Wiki (website collaboration for user groups)
  - TENA Issue Tracking System (task tracking system for user groups)
  - TENA Installer (cross platform software installation)
  - MagicDraw Plugin (converts UML diagrams in object model TDL syntax)
- **TENA Tools—Helping you conduct and manage your event**
  - TENA Middleware (C++, Java, .NET support for ~50 computer platforms)
  - TENA Console and Canary (event management and network monitoring)
  - DISGW (a TENA-to-DIS gateway).
  - ClearPath (multicast network testing)
  - TENA Data Collection System (collector, database export, and playback tools)
  - Interface Verification Tool (Platform generator to support testing activities)
  - Web Binding (provides JSON/REST http interface to TENA systems)
  - RelayNode (bridges different communication domains)
  - SIMDIS TENA Plugin (3D visualization and analysis support for TENA object models)
  - TENA Video Distribution System (various tools related to video/audio stream support)
  - Mission Information Resource Controller (automated configuration for distributed systems)
  - Network Communication Tools (chat, file transfer, etc.)

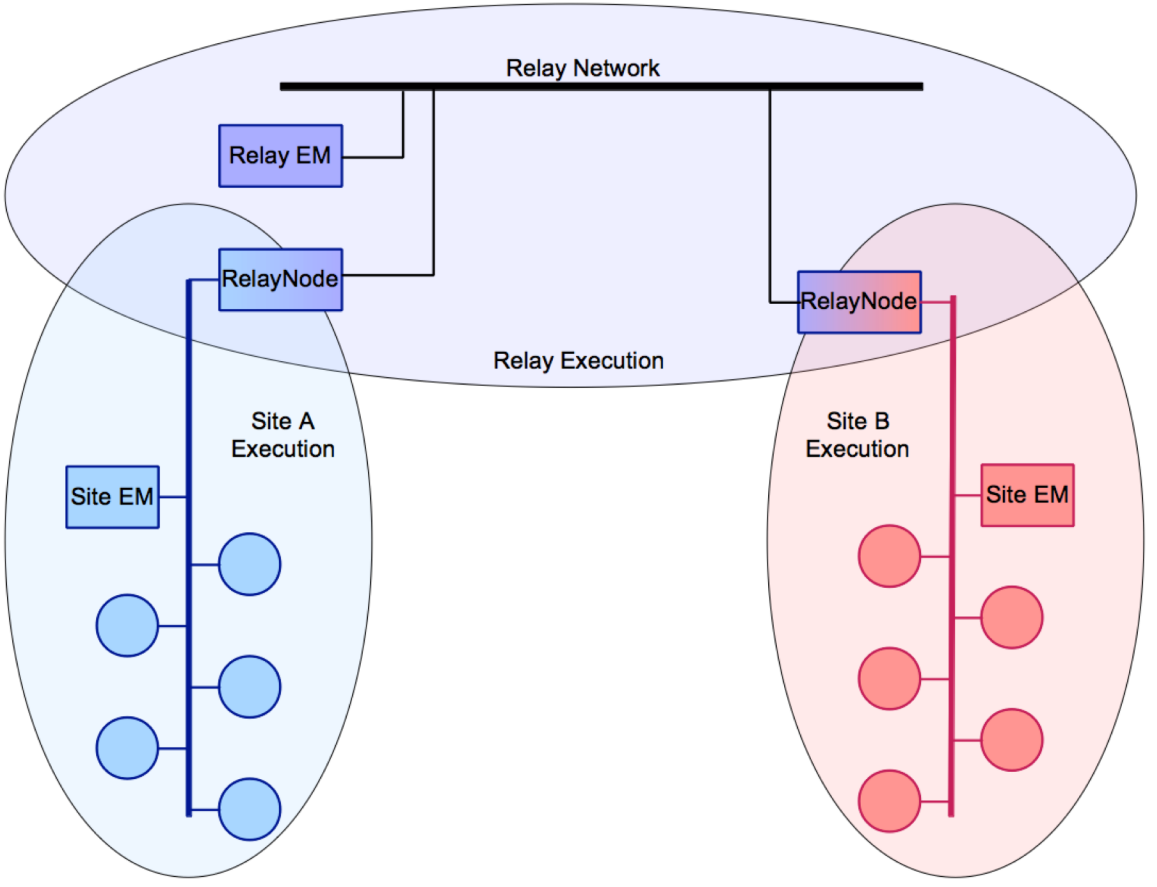
Other TENA Integration Methods: Web Binding

- **Web Binding is automatically generated based on object model**
- **Hub provides REST API to web clients to perform middleware operations (e.g., subscribe to type Vehicle and obtain updates)**
- **TENA data sent to and from the Hub uses JSON encoding**

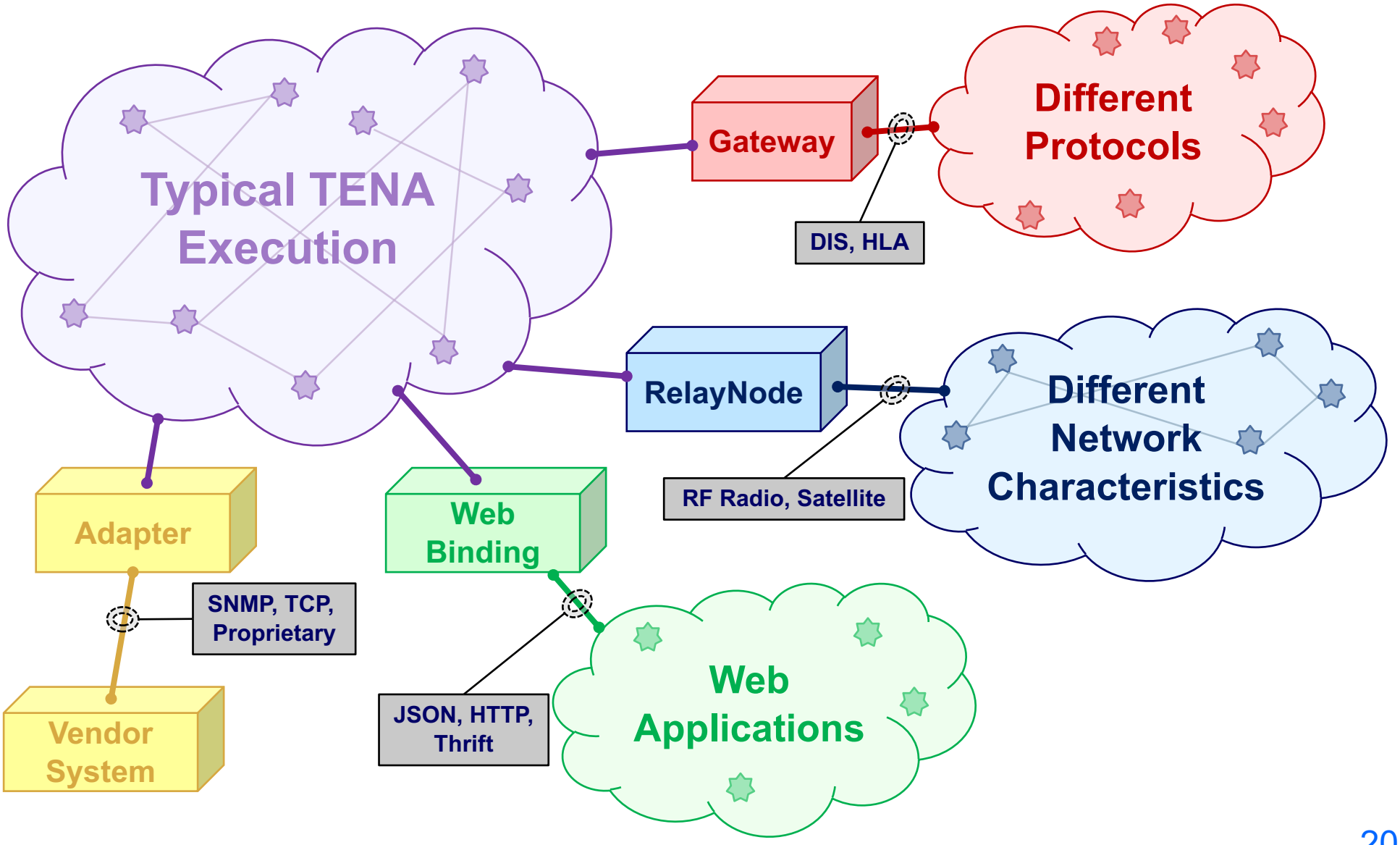


Other TENA Integration Methods: Relay Node

- **RelayNode is used to bridge two executions that may have different communication characteristics**
  - For example, one network segment may be for a low data rate link and an update would only need to traverse that link once, and then be replicated by the RelayNode to multiple subscribers
- **Illustration shows a typical scenario with a single WAN execution and multiple LAN executions**



TENA Interoperability Architecture Illustration



Summary of TENA/JMETC Capabilities

An Architecture for Ranges, Facilities, and Simulations to Interoperate, to be Reused, to be Composed into greater capabilities

- **A Working Implementation of the Architecture**
  - TENA Middleware currently works on Windows, Linux, and Sun
  - TENA Repository filled with information, tools, and object models
- **A Process to Develop and Expand the Architecture**
  - JMETC Users Group and AMT Meetings
- **A Technical Strategy to Deploy the Architecture**
  - JMETC process brings interoperability and reuse to test ranges
- **A Persistent Network to permanently connect test sites**
  - JMETC network enabled with TENA allows new tests to be performed with much less lead time and expense compared to the past

Contact Information

- **Both TENA and JMETC provide free technical assistance and training for any interested users. Contact information is below.**
- **TENA Website: http://www.tena-sda.org**
- **TENA Feedback: feedback@tena-sda.org**
- **JMETC Website: http://www.jmetc.org**

Live-Virtual-Constructive Interoperability Future Techniques

Randy Saunders

Agenda

- Why the future might be different
- What the future might include
- One Hypothetical

The Future ?

"Prediction is very difficult, especially if it's about the future."

-Niels Bohr

"The best way to predict the future is to invent it."

- Alan Kay

- The M&S community, as part of the greater software community, is carried along by emerging trends.
  - Open Source Collaboration
  - Multi-Purpose Data Streams
- Where might this lead?

Future Environment

Virtual/Constructive

Live

Users

Visualization

Applications

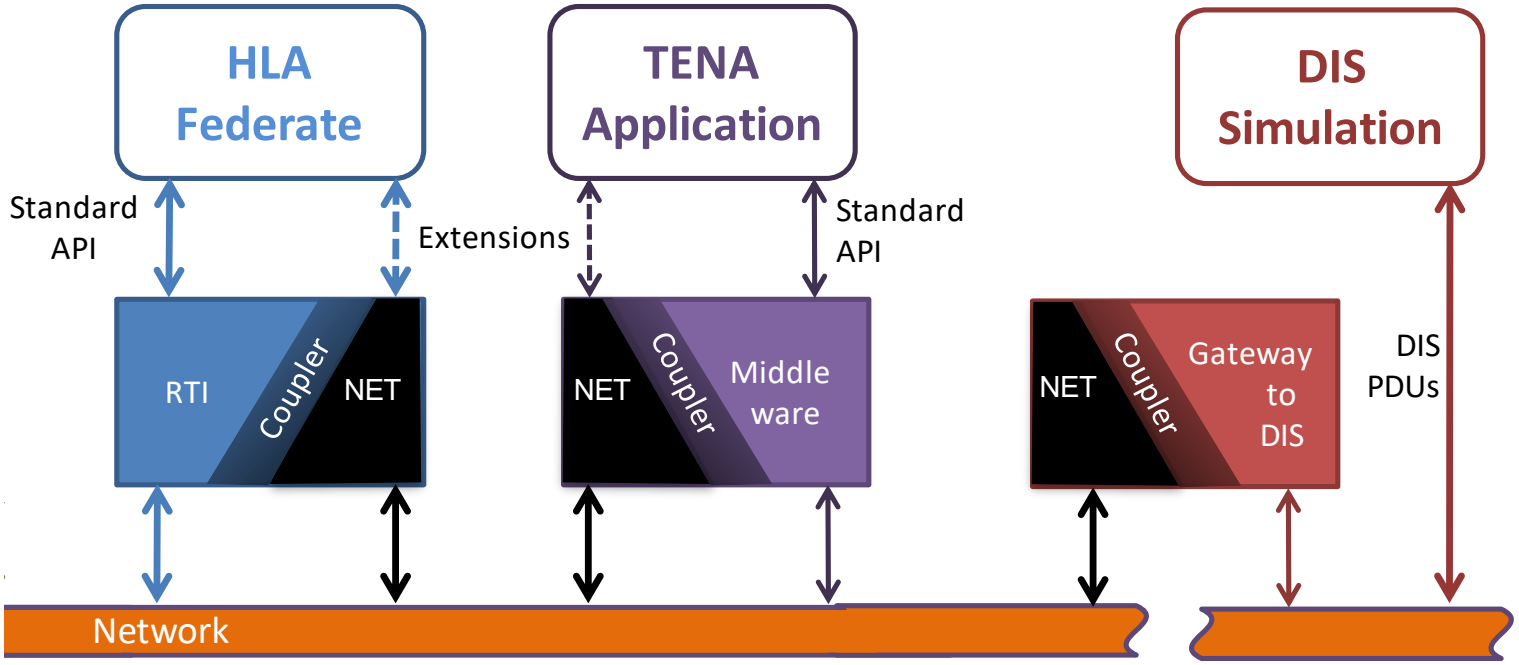
Simulation

Range

Who Really Cares?
All this is Off-The-Shelf to users.

APL

One (Potentially Random) Hypothetical



- Could an Open Source solution provide this sort of network?
  - DDS is an example that could.
  - Increasing use of DDS in the tactical community provides a reuse opportunity PLUS integration into military networks.

Cloud Computing, Simulation-as-a-Service, …

Users

Virtual/Constructive

Live

Visualization | Applications | Simulation | Range

Internet, Magic, Whatever …

Simulation #1 . Simulation #2 Service Provider Simulation #n

APL

Simulation in the Cloud

- Economic factors favor data center consolidation, virtualization, and cloud hosting for software solutions in general.

What about Simulation Interoperability?

- DIS is a wire standard, based on multicast UDP
  - None of the major commercial cloud providers support this.
- HLA and TENA are APIs, so the middleware vendor has to adopt some new communication solution for a VM environment.
  - DDS (used in command and control) may be available, some day
  - Service Interfaces of some form are mandated

All paths are TBD at this point.

Conclusion: What's the important Driver?

- Top Issues (From Ed's part of the intro):
  - Fair Fight
    - Terrain Correlation between different L/V/C sims when the sims interact with each other
    - Integrating live sensor systems with simulated entities (e.g., ACES)
    - Integrating live and simulated C4I (messages, voice, video)
  - Software Connectivity
    - Time synchronization (NTP? GPS? HLA/TENA "Time Management")
    - Bandwidth management (L/V/C interactions/messages within network limits)
    - Object Model and network transport protocol incompatibility
  - Hardware Connectivity
    - Bridging multiple security domains

- None of these issues is Architecture-Specific.
  - Different architectures provide different, small, subsets of the solution.
- Best-practice processes have been documented, including for multi-architecture situations.
  - You should use them!

Questions

Randy Saunders
Johns Hopkins APL
+1.443.778.3861
Randy.Saunders@jhuapl.edu

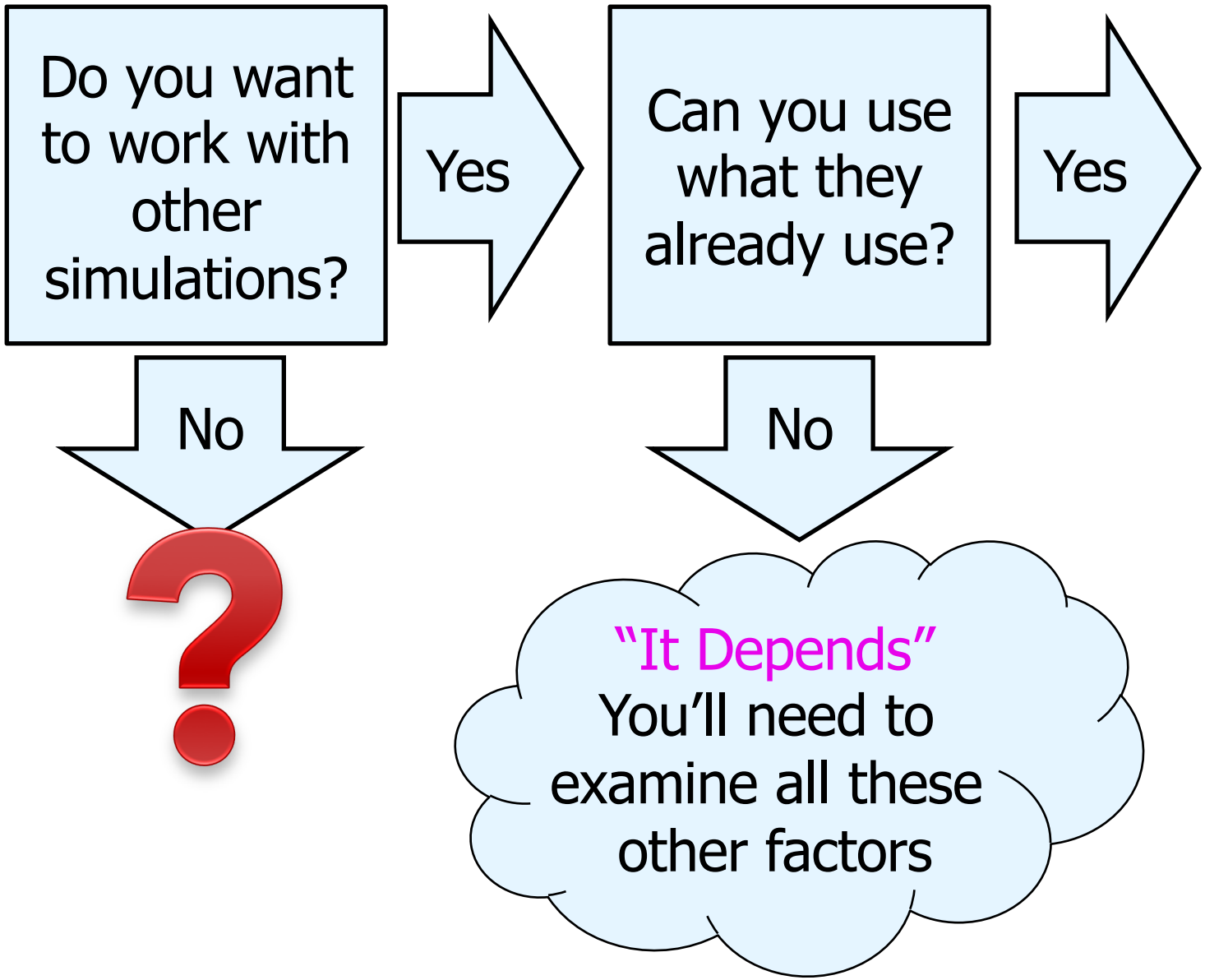
Dr. Edward T. Powell
Ed Powell Consulting LLC
+1.703.587.8036
epowell@tena-sda.org

Integration Architecture Advantages and Disadvantages

Outline

- **Algorithm for picking the right architecture**
  - Just Kidding
- **DIS – Distributed Interactive Simulation**
  - Pros and Cons
- **HLA – High Level Architecture**
  - Pros and Cons
- **TENA – Test/Training Enabling Architecture**
  - Pros and Cons
- **Real Suggestions for Your Events**

Algorithm for Picking the Right Architecture



DIS Advantages

- DIS can be used on any operating system, no matter how small or large, no matter how proprietary or embedded, how old or new.
- DIS software can be written in any computer language: C, C++, C#, Java, FORTRAN, Lisp, Ada, etc.
- DIS software can run on any computer hardware platform, from the tiniest embedded computer to the largest parallel supercomputer.
- The behavior of DIS packets on the wire is documented and well understood, relatively simple, and easily diagnosable.
- DIS packets are very bandwidth efficient.
- DIS is an international standard with support throughout the world.
- DIS has an extensive "standard object model" with many thousands of man-years of expertise behind it.
- When recovering from a network partition, the DIS "federation" is self-healing, although data collected from the partition period is corrupt.
- DIS does not require any centralized processes to operate properly, simplifying setup, testing, and operations.
- Software that implements the DIS standard is available in both commercial and open-source forms.

DIS Disadvantages

- DIS fails silently
- Relatively long integration time required
- (Mostly) fixed object definitions - limited extensibility
- Limited meta-model (C-struct-like PDUs) and model of the battlefield (entities and interactions between entities) is fixed.
- Dead reckoning giveth and dead reckoning taketh away
- Not optimized for WAN operations
- Paper standard - multiple interpretations likely, lots of code to write or acquire.
- Data definitions and data communication protocols are inextricably intertwined.
- Many mutually-incompatible extensions exist to meet operational needs beyond what DIS defines.
- Certain modeling constructs (e.g., EntityType) are modeled in a difficult way making it hard to change or work with.
- "My extensions to DIS are just fine; it's YOUR extensions that are screwing up the exercise!"

HLA Advantages (Part 1)

- Allows user-defined object models.
- Leverages the work done on DIS with a FOM based on DIS PDUs (RPR-FOM), though use of this FOM is not required.
- Supports both persistent objects ("objects") and messages ("interactions")
- Provides tested RTI interchangeability without code changes in the members.
- Provides the most complete set of standardized simulation services.
- Provides architectural support for variable time/global event ordering.
- HLA-2010 provides architectural support for detecting and addressing intermittent network upsets under software control without operator input (however, these services are not necessarily implemented by any given RTI).
- Provides mechanisms for bandwidth optimization through variable rate updates or data use subscription without the use of filters or bridges.
- HLA is an international standard.
- HLA software (RTIs) are available as commercial and open source packages.
- Many HLA-based tools and utilities are available either for sale or open source.

HLA Advantages (Part 2)

- Contains the capability to use a modularized FOM.
- Provides powerful interest management services between publishers and subscribers.
- Provides the ability to migrate ownership of an object between simulations
- Allows users to choose between best effort (multicast) and point-to-point reliable communication between federates on an object-class-by-object-class basis.
- Implementations can be extremely high performance (low latency, high throughput).

HLA Disadvantages

- Any given RTI works on only a limited set of Operating Systems/Compiler combinations
- No standardized wire format; indeed, the wire format is vendor-dependent and thus is completely opaque.
- RTIs from different vendors do not interoperate with one another.
- Meta-model (OMT) provides limited object composability.
- HLA objects are not "objects" in the object-oriented sense.
- Very complex API.
- No compiler-time error checking.
- Commercial RTIs may have substantial license costs.
- Most implementations require a centralized process of some sort to function.
- Architecture requires user-application marshaling and de-marshaling of data (though some implementations provide this for users).
- Many RTI implementations do not work properly on unreliable networks.

TENA Advantages (Part 1)

- Government owned, available to everyone free of charge
- Allows user-defined object models
- A robust standard object model exists (though using it is not required). It is not as extensive as DIS/RPR, but more relevant to the T&E range community.
- Includes robust auto-code generation capability that auto-generates federation-wide marshaling/de-marshaling code, links object models directly with application code, generates fully functional test applications, and provides users with almost-completely-filled-out auto-generated starting points.
- Supports both persistent objects ("SDOs") and messages ("messages").
- Contains an extremely robust fully object-oriented meta-model with many capabilities including object composition, distributed pointers, vectors, enumerations, remote method invocations, etc.
- Allows the standardization of both the interface to, and the implementation of, standard algorithms (local classes)
- Multiple standard spatial reference frames (SRFs—"coordinate systems") are built in to the standard TSPI object model, including seamless conversions between each.

TENA Advantages (Part 2)

- Type-safe interface eliminates entire categories of programming errors
- Allows users to choose between best effort (multicast) and point-to-point reliable communication between federates on an object-by-object basis.
- The Middleware is extremely high performance (low latency, high throughput).
- Supports an industry-wide reuse repository for object models, source code, and documentation.
- Many reusable tools and utilities exist for TENA, all free, including those for exercise management, data collection, and object-model management.
- Guarantees that users in a federation are using the identical object model definitions, and where appropriate, implementations.
- Numerous government information assurance approvals exist for the TENA software
- A commercial TENA-based cross-domain solution exists from secret to unclassified.
- TENA is actively supported with DoD funding for evolution and maintenance.

TENA Disadvantages

- Middleware works on only a limited set of Operating Systems/Compiler combinations.
- No standardized wire format (though a Wireshark plug-in does exist to give users some transparency in what is being transmitted).
- No ability to globally order events (i.e., does not support "HLA Time Management") or transfer of ownership of objects.
- Not an international (commercial) standard (but a government standard).
- The Middleware API requires object-oriented thinking by users (a disadvantage only to those users who think functionally).
- Limited tools exist for defining/deploying an event (except for OM tools).
- Software is not available as open source.
- Limited capability to function on unreliable networks.

Lessons for Your Events

- **Use the most capable interoperability architecture when judged by *your own* requirements**
- **Use multiple architectures connected with gateways *only* when that is the most cost-effective solution, *and* the use of the gateways will not compromise your event**
- **Remember that interoperability architectures only solve half the battle. Smart event design and knowledge of distributed communication issues are still required.**