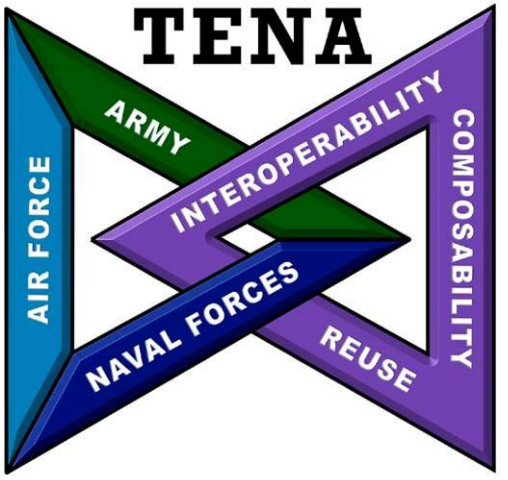
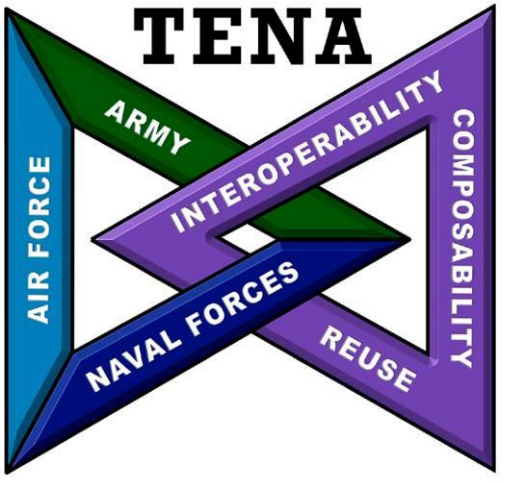


The Test and Training Enabling Architecture (TENA)—

Supporting the Decentralized Development of Distributed Applications and LVC Simulations

J. Russell Noseworthy, Ph.D.

TENA SDA Software Development Lead


TENA
ARMY
AIR FORCE
NAVAL FORCES
INTEROPERABILITY
COMPOSABILITY
REUSE

So Who Am I?


IMPROVING T&E
CTEIP
CAPABILITIES


UNITED STATES
JOINT FORCES
COMMAND
USJFCOM

Well, I'm Someone Who Knows Quite a Bit About HLA

1997 to 2000

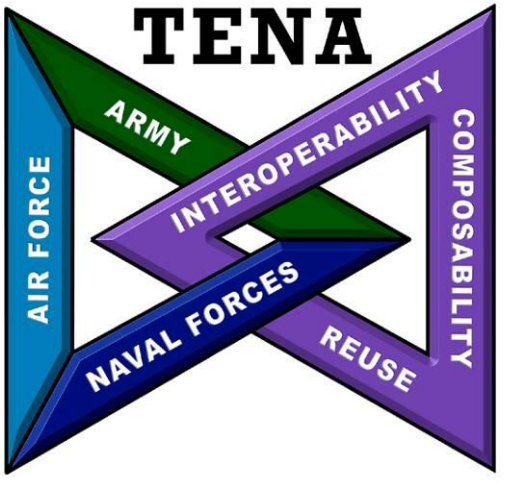
- **Led the design and development of the reference implementation of the DMSO HLA 1.3 RTI**
  - At the time, it was called RTI-NG (Later became known as the RTI-NG Pro)
  - The most widely-used RTI
    - Has supported simulations with hundreds of computers and a few hundred thousand entities
    - "Typical" Linux PC on a 100Mbps LAN:
      - Thousands of updates per second, sub-millisecond latency
- **Participated in the early days of IEEE HLA 1516 specification**

2000

- **Left the "HLA world" to create a better distributed system architecture**

2000 to Present

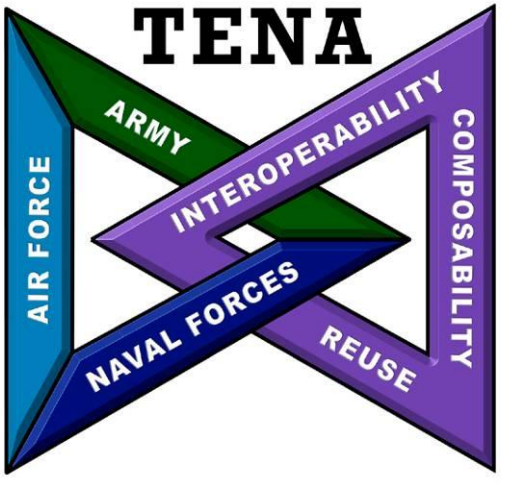
- **Software Development Lead for TENA SDA**



So What is TENA?

TENA Software Development Activity (SDA)

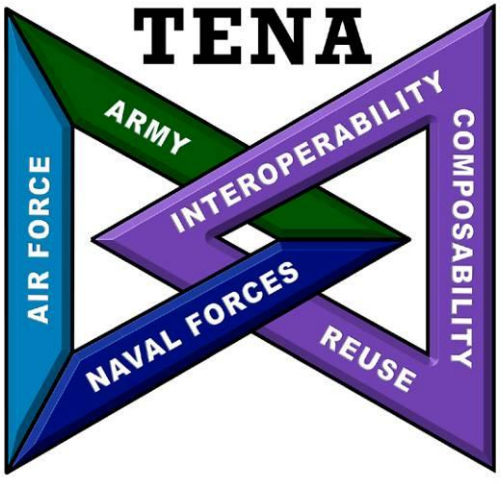
- **Test and Training Enabling Architecture (TENA)**
  - TENA is a long-term (~8 years so far) software infrastructure program that is transforming distributed computing in the DoD Testing and Training Community
  - Represents ~71 person years of technology development
  - Freely available (but a download account is required)
- **TENA is Applicable to Numerous Problem Domains**
  - TENA is agnostic to the particular user domain
    - The DoD Testing and Training community is the user domain "paying the bills"
  - TENA can be applied to distributed real-time high-speed synchronous collaboration problems, e.g.,
    - Controlling real-time collectors/data feeds and using advanced filtering techniques to efficiently disseminate information within a distributed network
    - Situational awareness systems that need to rapidly provide alert notifications and status information to various distributed viewer nodes
    - Distributed simulations and simulators



So Why is TENA Relevant to Distributed Simulation and Real-Time Applications?

Decentralized Development of Large-scale, Distributed, Real-time and Embedded Systems and the TENA SDA Project

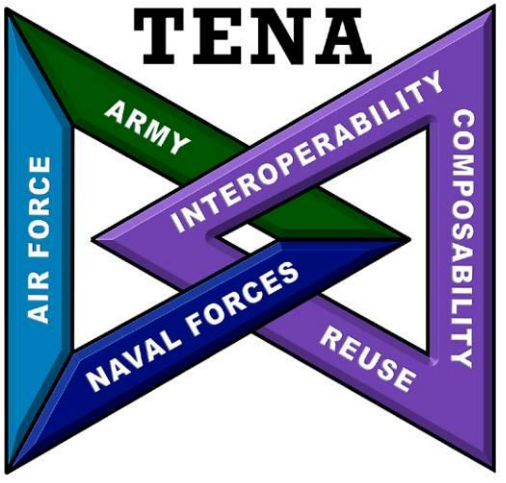
- DoD ranges use systems of sensors to take measurements for the purpose of testing and/or training.
- Many of the these sensor systems are embedded systems.
- The testing and training events occur in the real world. Thus, real missiles are launched, real tanks are driven, real planes are flown, etc.; and so real measurements must be taken in real-time.
- The sensor systems are themselves inherently distributed, typically over a large geographic area.
- The sensor systems can include half a dozen to several hundred individual component sensors.
- So, DoD ranges are large-scale, distributed, real-time and embedded (DRE) systems.
- The developers working on these systems are themselves geographically distributed, may have never met each other, and have no common authority (e.g., different companies, different services, different countries)
- The TENA SDA project is intended to support DoD ranges.
- Thus, the TENA SDA project must support distributed and decentralized development of large-scale DRE applications.



That Covers Distributed and Real-Time, But What About Simulation?

TENA and Distributed Simulation

- **As a general rule, TENA is a better choice than the HLA/RTI to implement most distributed simulations**
  - More on that later …
- **Two exceptions to that general rule:**
  - At the moment, TENA doesn't provide an implementation of time management
  - At the moment, TENA doesn't provide an implementation of a "game pause" or synchronization points
- **Why not?**
  - The customers sponsoring the development of TENA aren't particularly interested in those features at this time.
    - They fire real missiles, and real missiles don't pause in mid-flight or respect simulation time
- **There is one class of simulation that is especially relevant to TENA …**



Three Letters: LVC

What is LVC?

- **Live-Virtual-Constructive (LVC) System**
  - Live—Real (i.e., not simulated) physical entities (e.g., a real plane)
  - Virtual—Simulators, i.e., virtual environment emulating real physical entities operated by a human (e.g., a plane simulator)
  - Constructive—Purely synthetic world where arbitrarily large numbers of entities interact based on (complex) models (e.g., a war game)
- **Example LVC System**
  - Pilot of a real fighter jet
  - Pilot in a fighter jet simulator acting as wingman to real fighter jet
  - Both engaging completely synthetic enemy fighters
- **The Desire for LVC Systems Has Greatly Increased Recently**
- **LVC Systems Are Not Well-Understood**
  - Lot's of opportunities for research to improve our ability to build and characterize such systems!

That's Lvc—With a Big L

- **For LVC—the demands of the Live components drive the rest of the system**
  - Failure is Not an (Inexpensive) Option!
- **LVC Systems are Made up of Heterogeneous Applications**
  - Developed at different times, in different places, by different people

Formal, *computer-enforced* agreements describing the nature and form of data exchanged in a large-scale LVC system are necessary to provide a common understanding of how and what data is to be communicated and furthermore, to ensure that that understanding is then implemented in every application comprising the system.

How Does TENA Address the Challenges of Large-scale DRE Systems?

Successfully developing applications for large-scale DRE systems is very difficult for most (all?) programmers.

Decentralized development makes this greatly complicates this already difficult task.

Reliability, maintainability, and understandability are critical components for success

TENA provides:

- **Model-based, high-level programming abstractions.**
- **Bug prevention through compile-time type checking and an API that's hard to use wrong.**
- **Model-driven code generation of custom-tailored core middleware software.**
- **Complete, working, model-based applications, ready for customization by programmers.**

The TENA Middleware uses model-driven automated code generation to reduce the amount of software that must be written (and tested) by humans. Furthermore, the TENA Middleware provides the application developer with a powerful programming abstractions. These programming abstractions are easy for the application developer to understand, resulting in applications with fewer mistakes.

Combining Paradigms to Create the TENA Middleware

- **TENA Middleware combines distributed shared memory, anonymous publish-subscribe, and model-driven distributed OO programming paradigms into a single distributed middleware system.**
- TENA Middleware provides high-level abstractions using models to drive the automatic code-generation of complex distributed applications.
- TENA Middleware offers programming abstractions not present in HLA and provides a strongly-typed API that is much less error-prone than the HLA API.
- Reduces programming errors and enables developers to quickly and correctly express the concepts in their applications.
- Re-usable standardized object interfaces and implementations further simplify application development.

The Ways in which TENA Applications Can Communicate

TENA provides to the application developer a unification of several powerful inter-application communication paradigms:

- **Publish/Subscribe**
  - Each application publishes certain types of information to which any other application can subscribe
  - Similar in effect to HLA, DIS, CORBA Event Service, NDDS, etc.
- **Remote Method Invocation (RMI)**
  - Each object that is published may have methods that can be remotely invoked by other applications
  - Similar to CORBA RMI or Java RMI
- **Distributed Shared Memory (DSM)**
  - Applications read and write the state of objects as if they were local objects, even though they are remote objects
  - A very natural, easy to understand programming paradigm that projects the illusion of working on a shared memory multi-processor machine onto a distributed computing system
- **Messages**
  - Individual messages that can be sent from one application to other applications

The Stateful Distributed Object: SDO

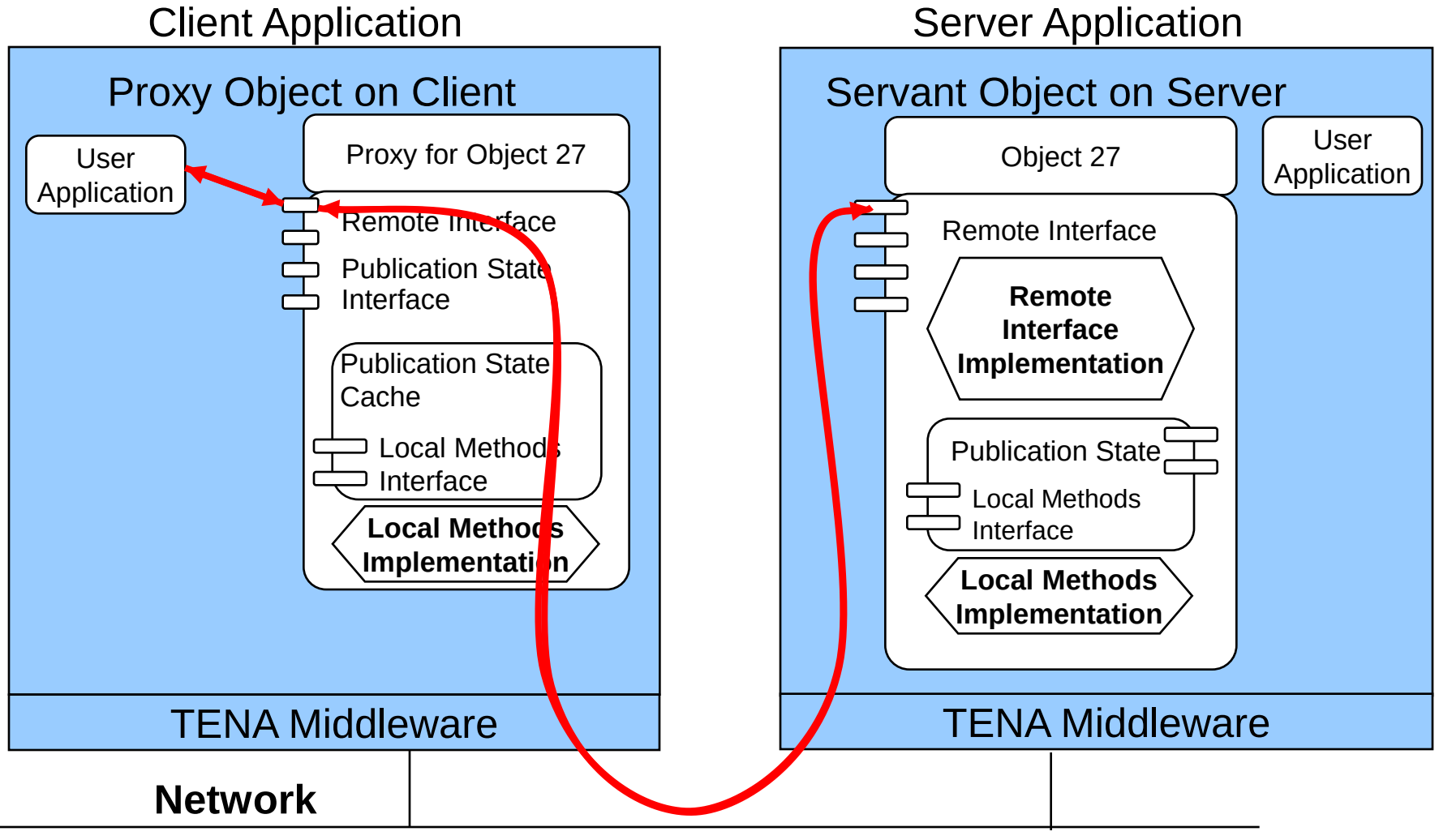
- **A Stateful Distributed Object SDO is an abstract concept formed by the combination of a distributed object interface with data or *state*. The state is data attributes of the SDO that are disseminated via publish-subscribe and cached locally at each subscriber.**
  - An SDO supports the remote method invocation concept that is very natural to distributed object-oriented system programmers.
  - An SDO provides direct support for disseminating data from its source to multiple destinations.
  - An SDO supports reads and writes of data as if it were any other local data—a concept familiar to virtually every modern programmer.
  - An SDO's model-driven automatically generated code eliminates the tedious and error-prone programming chores common to distributed programming.
  - An SDO's API is easy to understand and **hard to use wrong**.

Some other Key Constructs in the TENA Metamodel

- **A Local Class is similar to an SDO in that it too is composed of both methods and attributes. However the methods and attributes of a local class are always local with respect to the application holding an instance of the local class.**
- **A Message is a local class that can be directly disseminated to subscribers.**
- **An SDO Pointer behaves pretty much the same as pointers to objects in C++.**
- **SDOs, Local Classes and Messages all support Inheritance and Containment**
  - Dissemination of SDO updates and Messages follow the implied behaviors of inheritance and containment

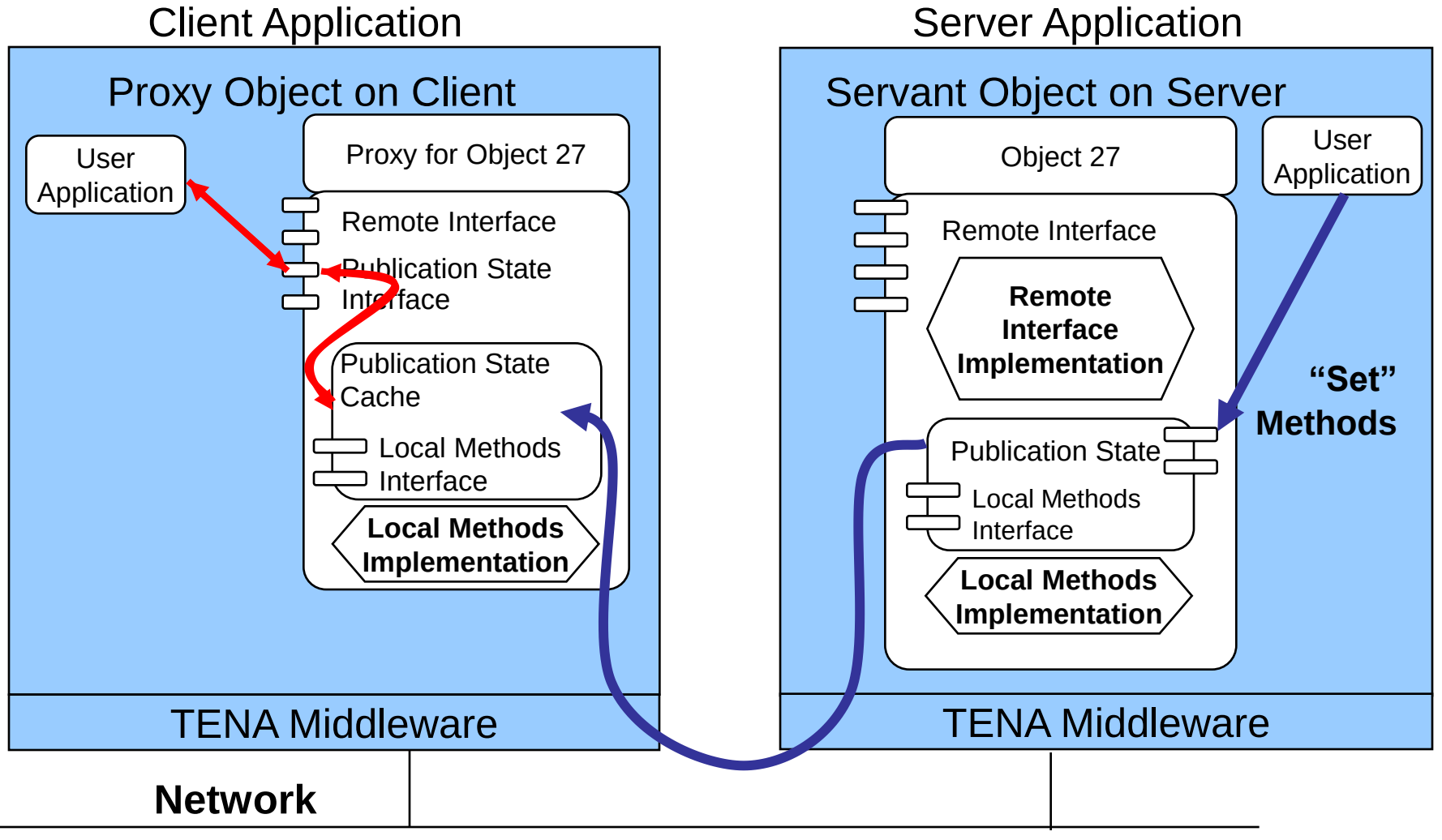
Clients and Proxies, Servers and Servants

- **Remote Method Invocation**
  - Work always performed on the server



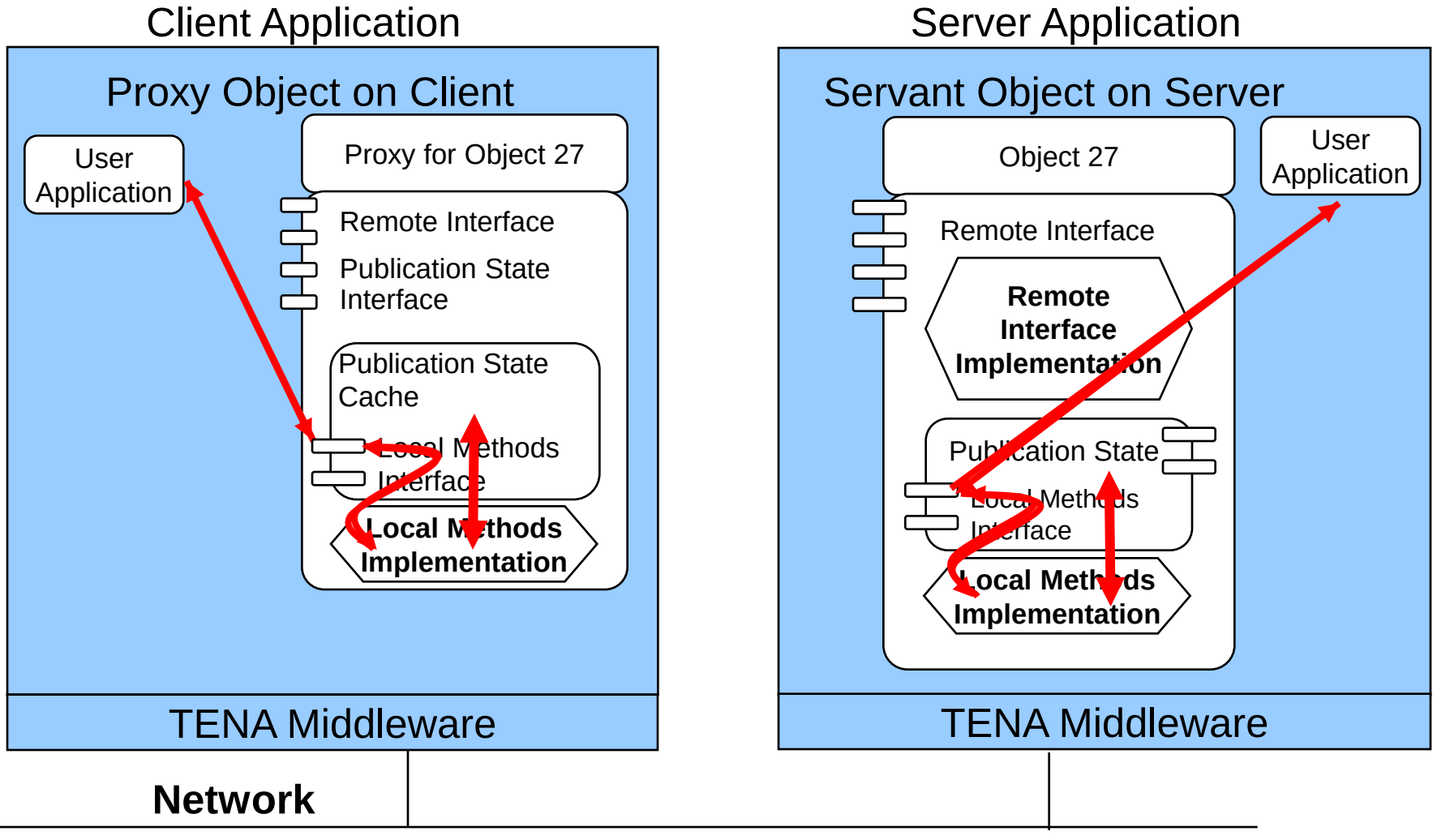
Clients and Proxies, Servers and Servants

- **Publication State Dissemination and Access**



Clients and Proxies, Servers and Servants

- **Local Methods used on both Client and Server**
  - Always performed locally on either client or server



Example TENA Object Model in TENA Definition Language (TDL)

```
package Example
{
 local class Location
 {
  float64 distanceFrom( in Location here );
  float64 xInMeters;
  float64 yInMeters;
  optional float64 zInMeters;
 };
 local class Date
 {
  Date(); // Default to today
  Date( int16 year, uint8 month, uint8 dayOfMonth );
  readonly int16 year;
  readonly uint8 month;
  readonly uint8 dayOfMonth;
 };
 class PhysicalThing
 {
  void moveTo( in Location newLocation );
  Location location;
  optional float64 massInKilograms;
 };
 class Person : extends PhysicalThing
 {
  string firstName;
  optional string middleName;
  string lastName;
  const Date dateOfBirth;
 };
 class Vehicle : extends PhysicalThing
 {
  boolean loadPassenger( in Person * pPassenger );
  boolean unloadPassenger( in Person * pPassenger );
  void driveTo( in Location newLocation );
  const string licensePlate;
  Person * pDriver;
  vector < Person * > passengers;
 };
 message TrafficReport
 {
  float64 distanceFrom( in Location here );
  readonly optional Location troubleSpot;
  readonly string report;
 };
};
```

Updating an Example::Person SDO's State

To change an SDO's state, updaters allow sets of attributes to be modified atomically

```
std::auto_ptr< Person::PublicationStateUpdater >
  pUpdater( pPerson->createUpdater() );

pUpdater->set_location( Location::create( 1.2, 3.4 ) );
pUpdater->set_massInKilograms( 3.14159 );
pUpdater->set_firstName( "Russ" );
pUpdater->set_lastName( "Noseworthy" );

pPerson->commitUpdater( pUpdater );
```

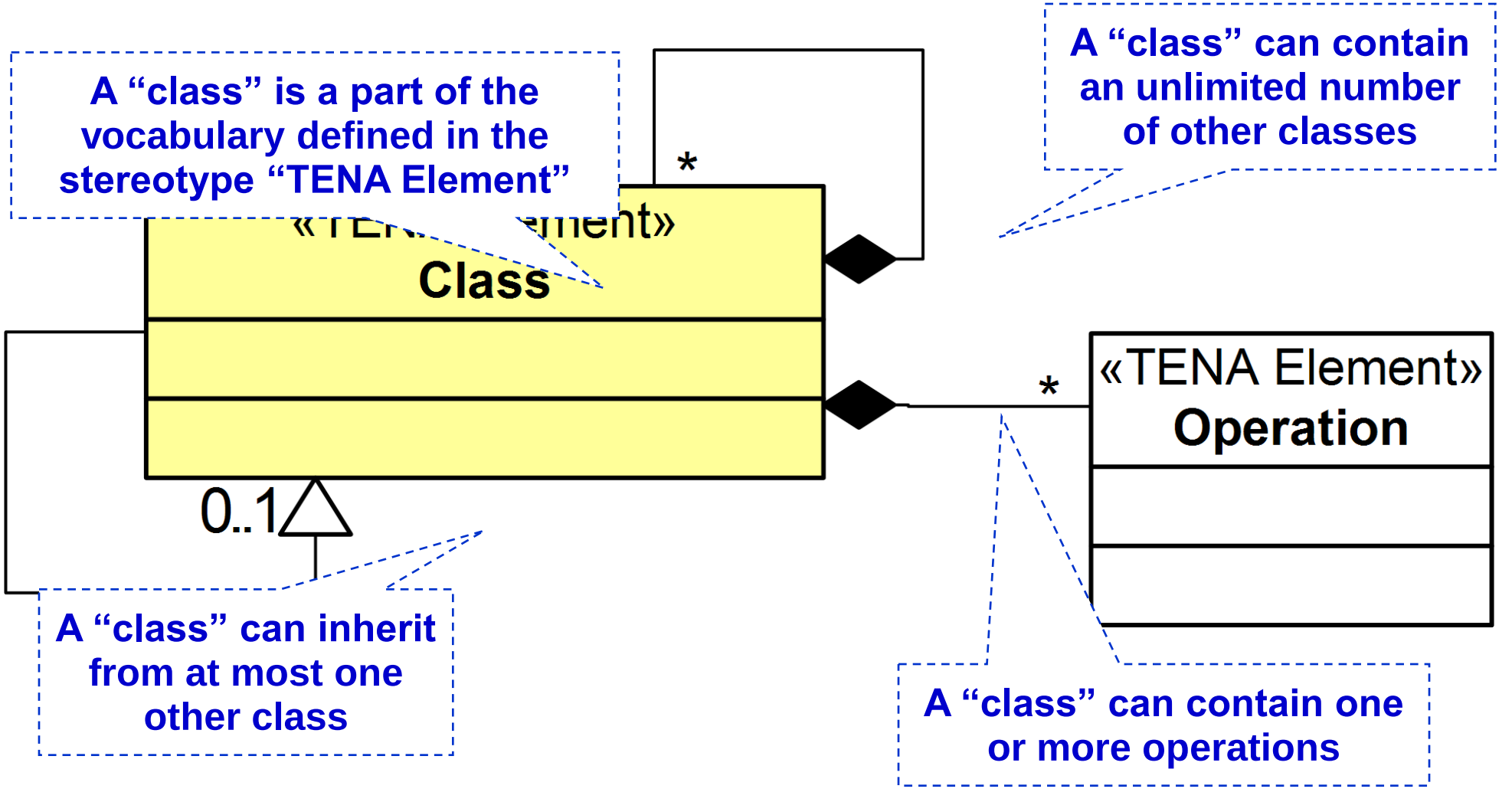
Every Computer Language Has A Meta-Model

(…and They're All Different)

- **C++**
  - Classes, structs == classes, abstract base classes, multiple inheritance, composition, generics, functions, methods, operators, fundamental types, exceptions, arrays, etc.
- **Java**
  - Classes, interfaces, exceptions, etc.
  - No structs, no functions, no multiple inheritance
- **CORBA IDL**
  - Interfaces, structs, valuetypes, sequences, enumerations, multiple inheritance of interfaces, unions, etc.
  - No classes
- **HLA**
  - HLA Classes ("objects"), interactions, attributes, single inheritance
  - No interfaces, no composition, no functions/methods, no local objects, no ...

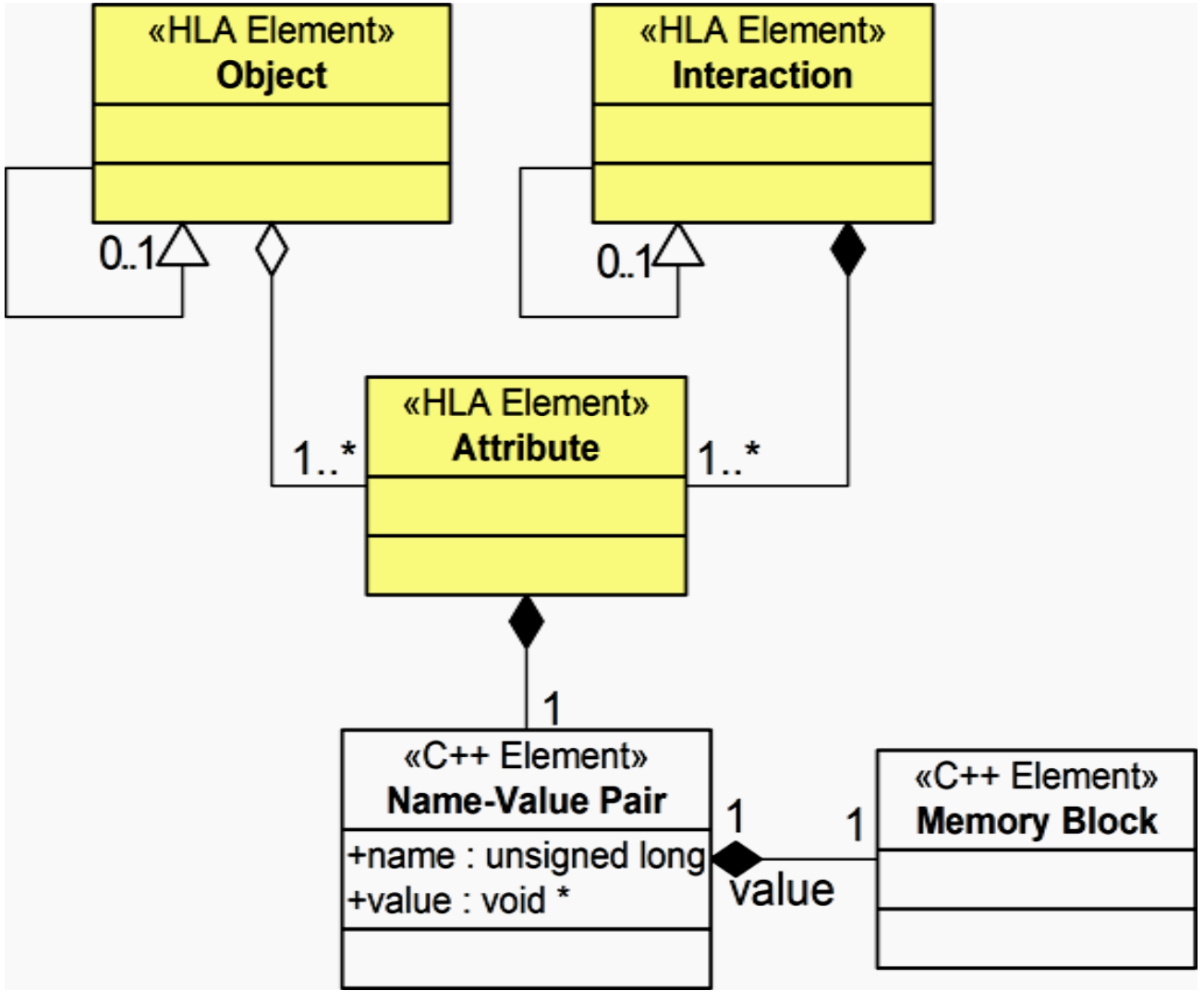
Representing a Meta-Model

- **"Pseudo-UML" is used, since formal UML is not as compact or communicative**



HLA Meta-Model
(with C++ additions)

- **Based on the HLA Object Model Template (OMT)**

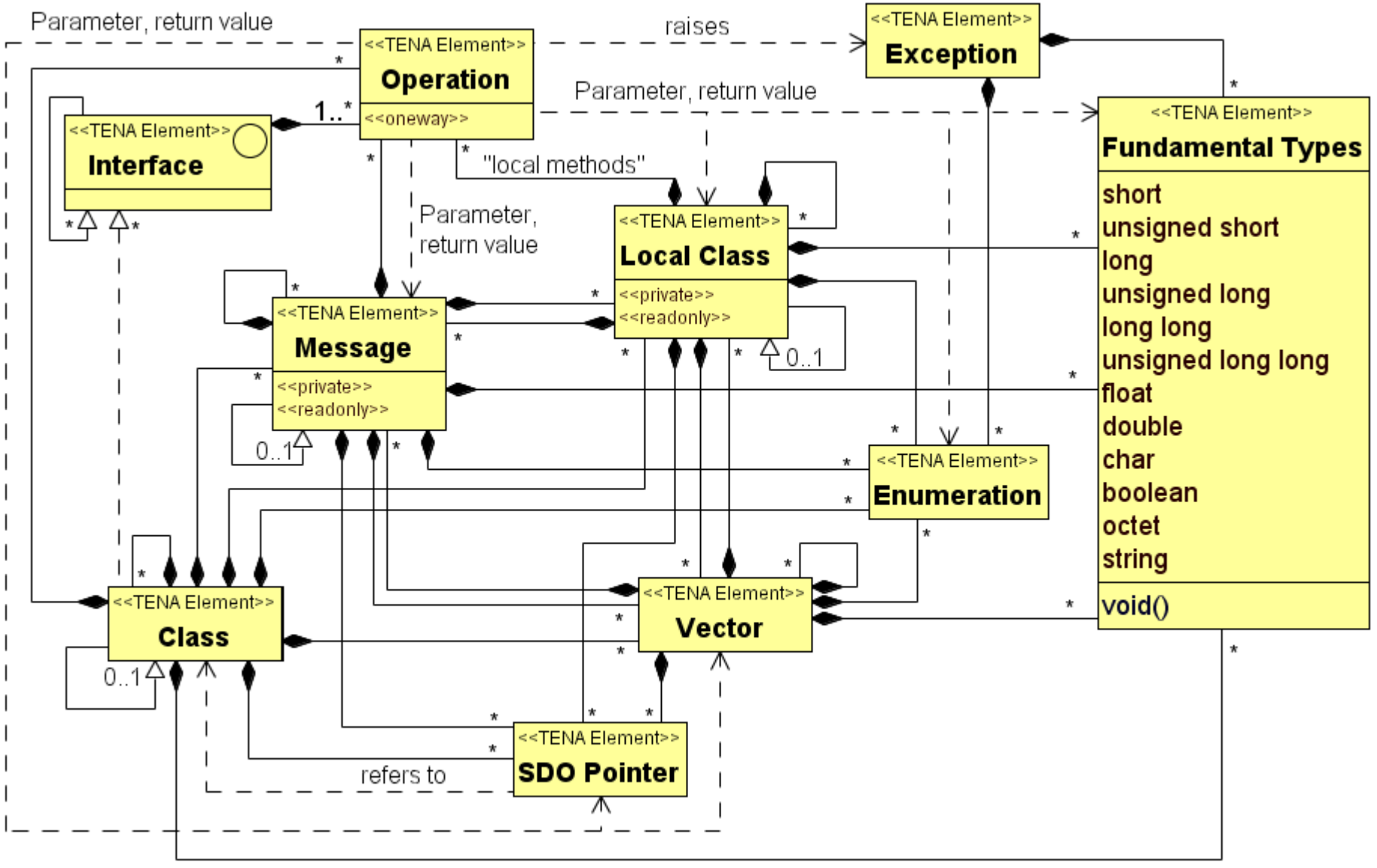


= may extend/inherit from

= may contain

= may loosely contain

TENA Meta-Model Release 5.2.2

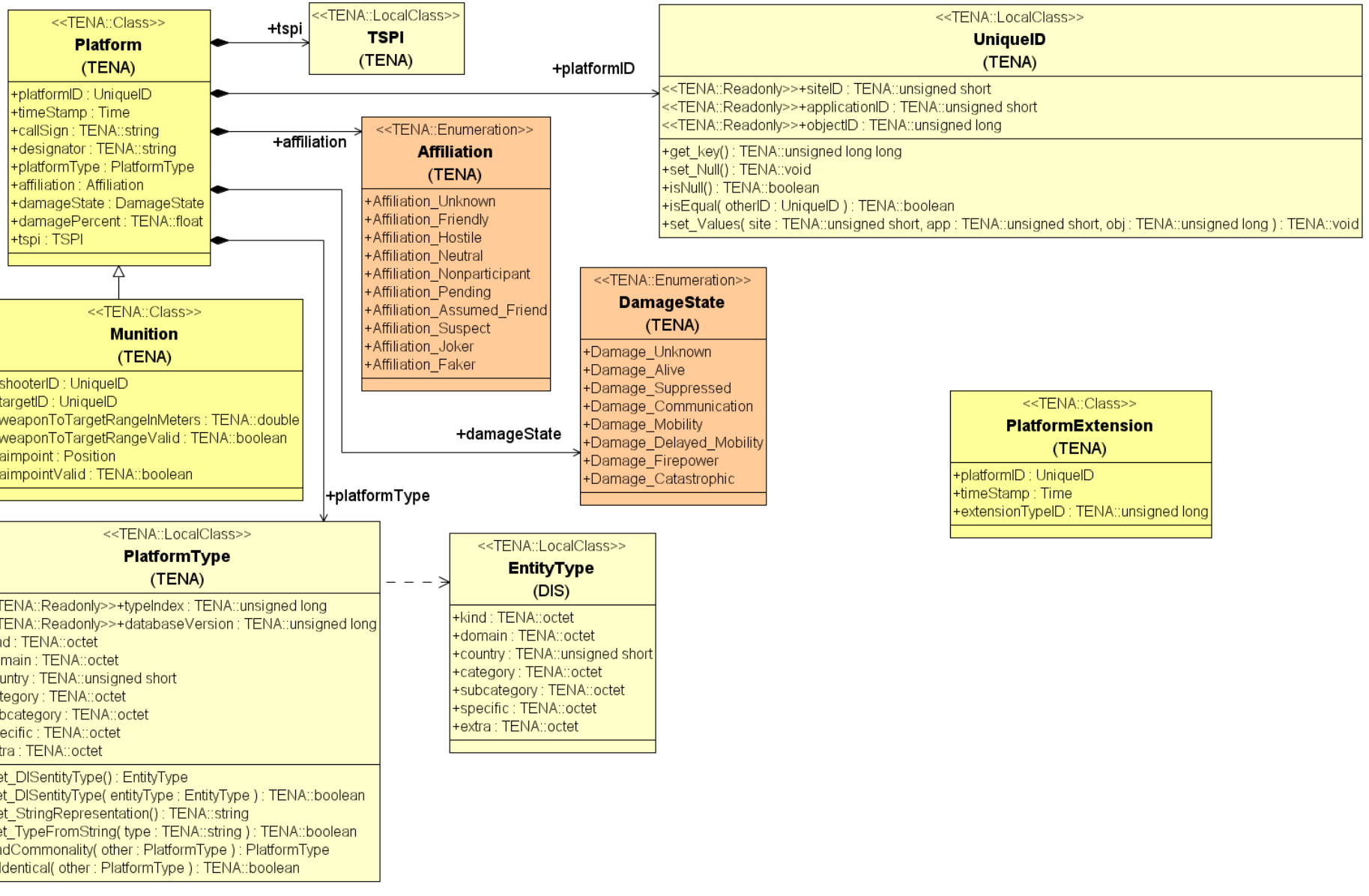


Object Models Developed and Maintained by the TENA SDA

- **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-v2
  - TENA-EmbeddedWeapon-v2
- **TENA-AMO:**
  - TENA-AMO-v1
- **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-Radar-v2**
- **TENA-GPS-v2**

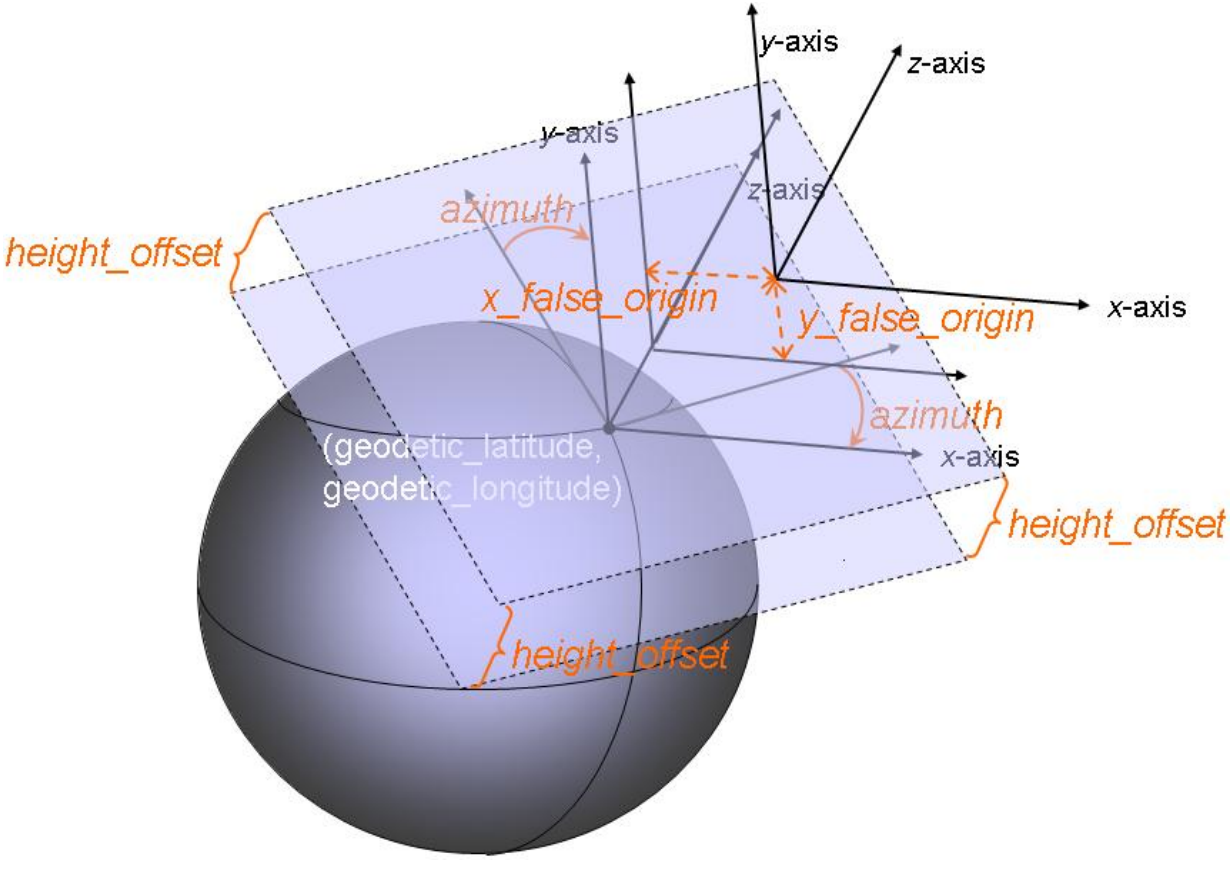
The OMs in bold also have behavior implementations developed and maintained by the TENA SDA project.

TENA-Platform-v3.1



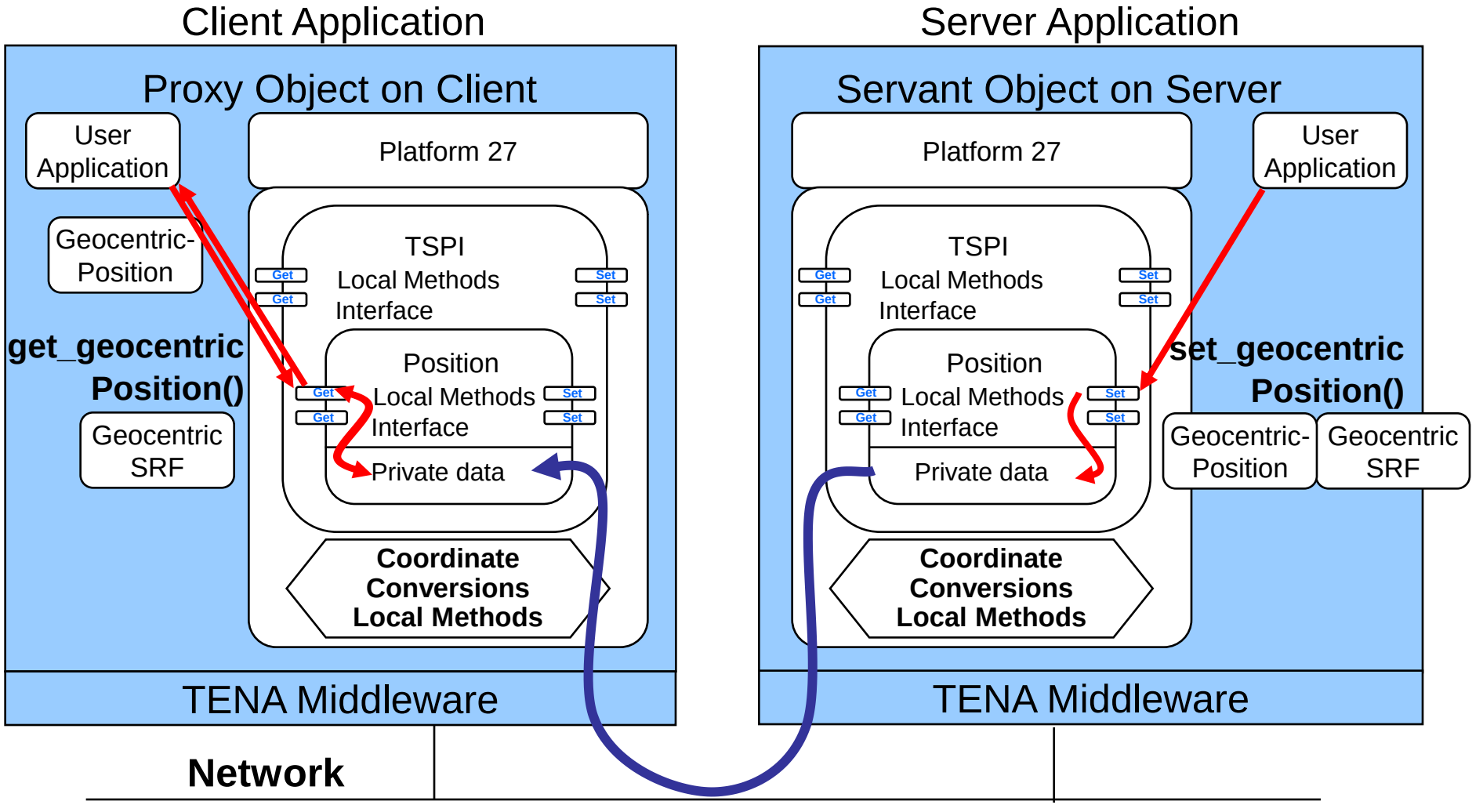
Coordinate Conversions
The TENA-TSPI Object Model

- **Which way is up?**
  - For geographically large events, e.g., missile tests, the answer is not obvious!
- **Not everyone uses the same means to measure position (and orientation)**
- **The TENA SDA provides Coordinate Conversion software packaged up in TENA local classes**
  - Provides for easy re-use of complicated software and abstraction of complicated problems



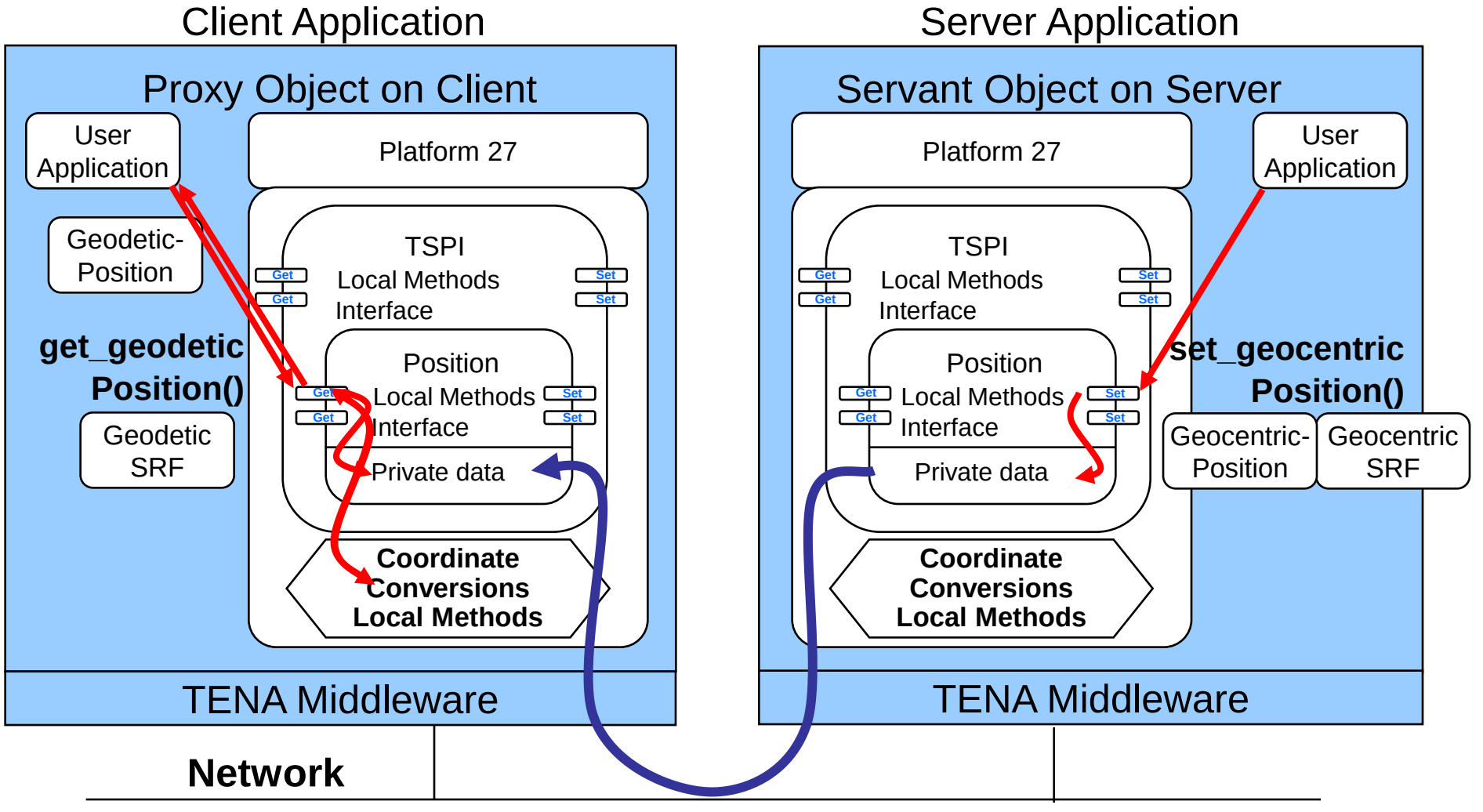
TSPI v4 with Coordinate Conversions

- **Case 1: Reading and writing in the same coordinate system**



TSPI v4 with Coordinate Conversions

- **Case 2: Reading and writing in different coordinate systems**
  - Write in Geocentric (ECEF), read in Geodetic (latitude/longitude/altitude)

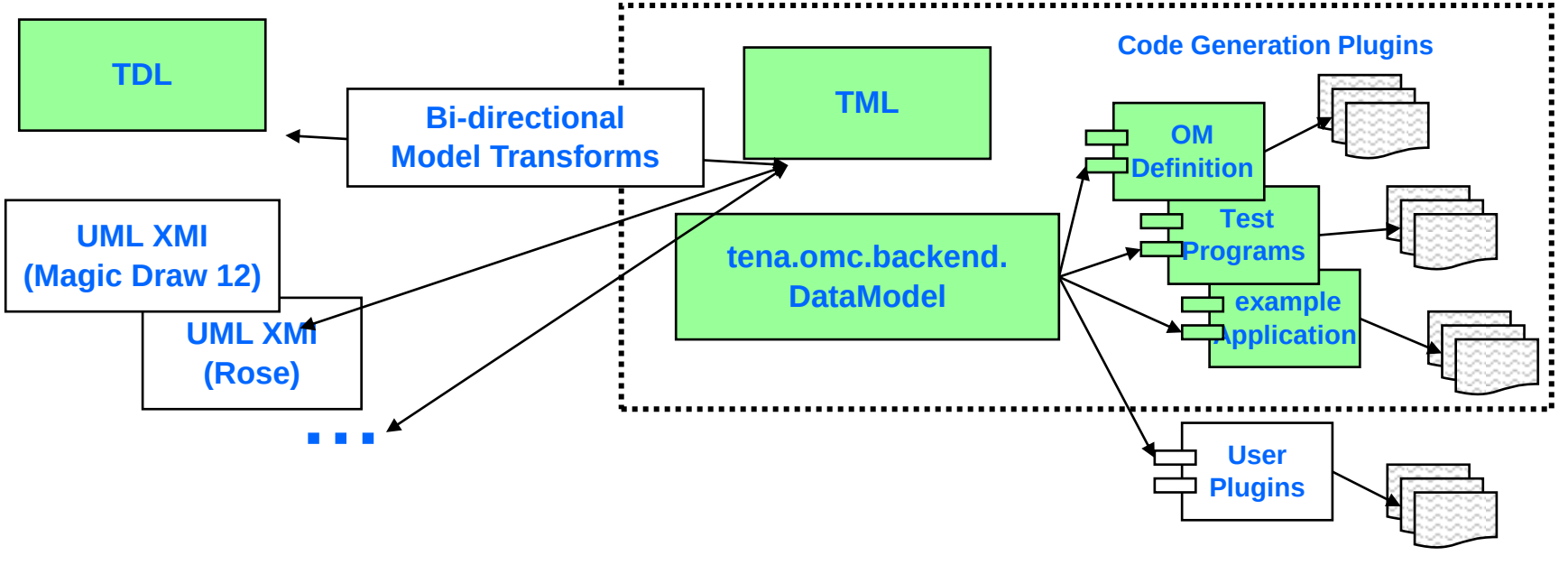


TENA Object Model Compiler Automatic Code Generation

- **TENA uses auto-code generation to provide the high-level programming abstractions of the TENA meta-model**
- **The TENA OM Compiler is a Java Application**
  - Has an extensible plugin architecture
- **The Meta-model is the key …**
  - User's can auto-generate "anything" from a TENA OM
    - Database logging applications
    - Visualization applications
    - Test applications
    - Gateways to other systems
      - A Generic gateway builder already exists
        - Supports DIS and HLA
    - Bindings to other programming languages
      - Java, .Net, MATLAB have all been done …

Auto Code Generation with the TENA Object Model Compiler

- **The TENA Object Model Compiler is a Java application**
  - API and framework being developed to support various “code generation plugins” used to automatically create specialized code based on user-supplied FreeMarker templates



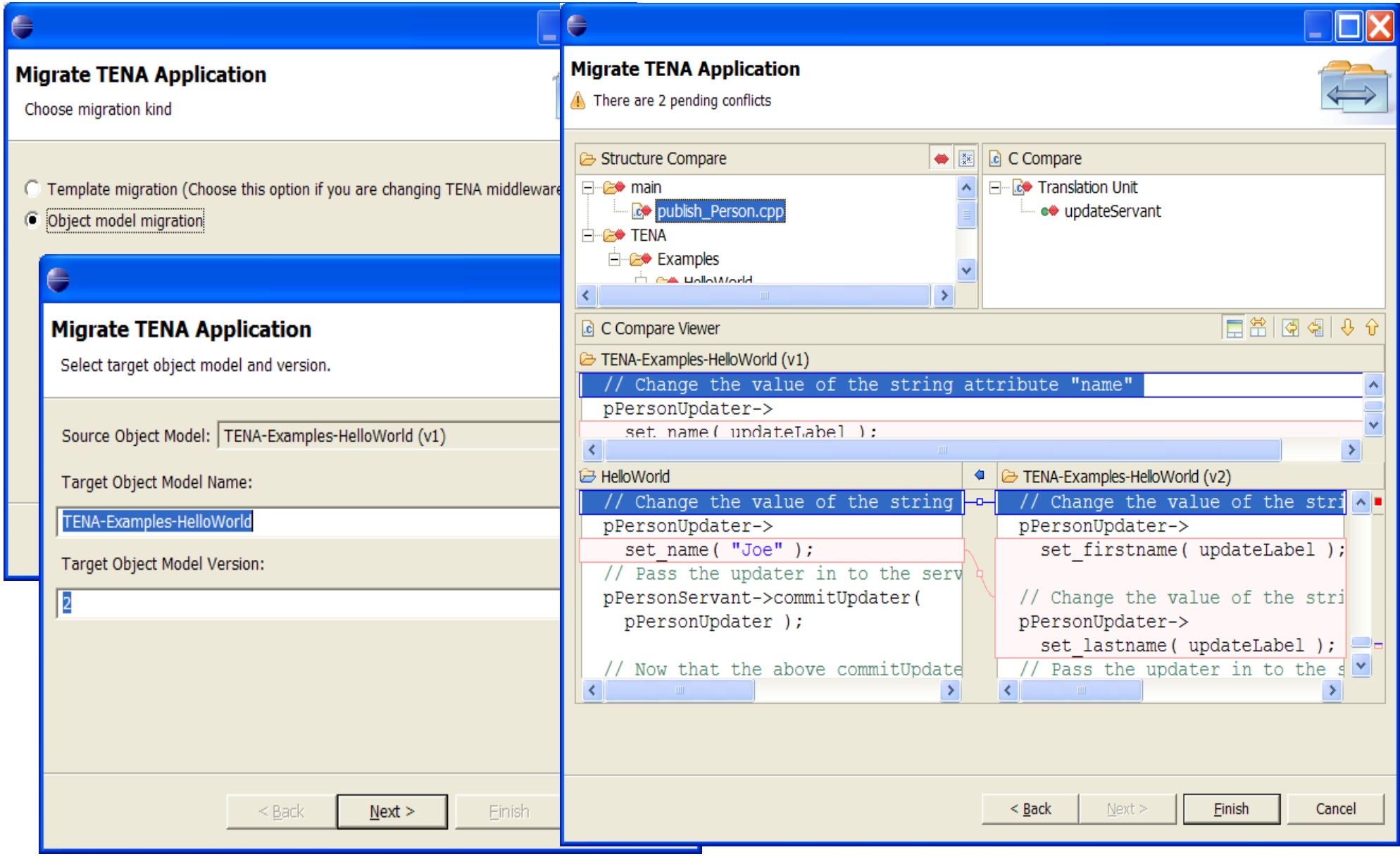
TENA Integrated Development Environment (TIDE)

- **TENA IDE based on eclipse**
- **Understands the TENA-Metamodel**
- **Can run the TENA OM Compiler to generate example applications or anything else the TENA OM Compiler can generate**
- **Supports source code migration to upgrade applications to new versions of the TENA Middleware**
- **Assists source code translation of HLA/RTI applications to TENA**
  - The (nearly typeless) RTI API hampers the ability to automate this process
- **Supports source code migration to update applications when an OM changes**
  - Uses a “Change Models” to model the changes to models

Example TENA Object Model in TENA Definition Language (TDL)

```
package Example
{
 local class Location
 {
  float64 distanceFrom( in Location here );
  float64 xInMeters;
  float64 yInMeters;
  optional float64 zInMeters;
 };
 local class Date
 {
  Date(); // Default to today
  Date( int16 year, uint8 month, uint8 dayOfMonth );
  readonly int16 year;
  readonly uint8 month;
  readonly uint8 dayOfMonth;
 };
 class PhysicalThing
 {
  void moveTo( in Location newLocation );
  Location location;
  optional float64 massInKilograms;
 };
 class Person : extends PhysicalThing
 {
  string firstName;
  optional string middleName;
  string lastName;
  const Date dateOfBirth;
 };
 class Vehicle : extends PhysicalThing
 {
  boolean loadPassenger( in Person * pPassenger );
  boolean unloadPassenger( in Person * pPassenger );
  void driveTo( in Location newLocation );
  const string licensePlate;
  Person * pDriver;
  vector < Person * > passengers;
 };
 message TrafficReport
 {
  float64 distanceFrom( in Location here );
  readonly optional Location troubleSpot;
  readonly string report;
 };
};
```

Migrating TENA Projects between object model releases



Architecture Management Team (TENA AMT)

Meetings every 3 months

AMT-40 Dec 18-19 2008 in Austin, Texas

- **AMT Members:**
  - 329 Armament Systems Group (329 ARSG)
  - Aberdeen Test Center (ATC), Aberdeen Proving Ground, MD
  - Air Armament Center (AAC), Eglin AFB, FL
  - Air Force Flight Test Center (AFFTC), Edwards AFB, CA
  - Army Operational Test Command (OTC), Fort Hood, TX
  - Common Training Instrumentation Architecture (CTIA)
  - Dugway Proving Ground (DPG)
  - Electronic Proving Ground (EPG)
  - integrated Network Enhanced Telemetry (iNET)
  - Interoperability Test and Evaluation Capability (InterTEC)
  - Joint Fires Integration & Interoperability Team (JFIIT)
  - Joint National Training Capability (JNTC)
  - Naval Air Warfare Center – Aircraft Division
  - NAWC – Weapons Division
  - Naval Aviation Training Systems Program Office (PMA-205)
  - Naval Undersea Warfare Center (NUWC)
  - NAVSEA Warfare Center - Keyport
  - P5 Combat Training System (P5CTS)
  - Pacific Missile Range Facility (PMRF)
  - Redstone Technical Test Center (RTTC)
  - T&E/S&T Non-Intrusive Instrumentation
  - White Sands Missile Range (WSMR)

Advising Members:

- **BMH Associates, Inc.**
- **Boeing**
- **Cubic Defense**
- **DRS**
- **Embedded Planet**
- **EMC**
- **Kenetics**
- **MAK Technologies**
- **NetAcquire**
- **Science Applications International Corporation (SAIC)**
- **Scientific Research Corporation (SRC)**
- **Scientific Solutions, Inc. (SSI)**

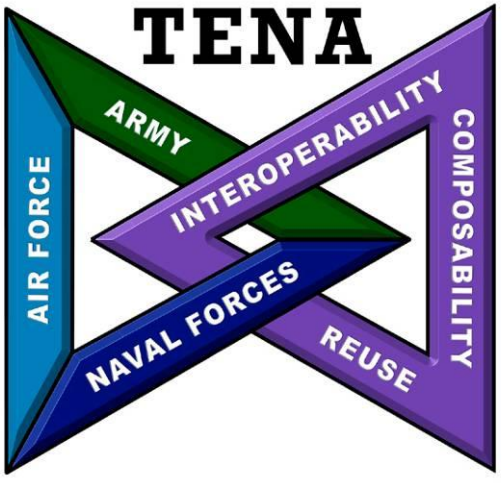
- **Design Decisions / Trade-offs / Status / Technical Exchanges of Lessons Learned / Use Cases / Testing / Issues & Concerns Identification, Investigation & Resolution**

Contact Information

- **Project Website: https://www.tena-sda.org/**
- **Download TENA Middleware**
  - **https://www.tena-sda.org/repository/**
    - Get the Beta version of Release 6
      - Don’t bother with Release 5
        - It’s about to become obsolete
- **Submit Helpdesk Cases**
  - **https://www.tena-sda.org/helpdesk/**
  - Use for questions about the Middleware

- **Feel free to contact me:**

J.Russell.Noseworthy@TENA-SDA.org



Questions or Comments?

TENA Architecture Overview