



The  
Test and Training Enabling Architecture  
(TENA)  
and Its Use of the  
SEDRIS Spatial Reference Model (SRM)



*Ed Powell*  
TENA Architect





# Agenda



- **TENA Overview and the Logical Range concept**
- **TENA Meta-Model – SDOs and Local Classes**
- **Using the SEDRIS SRM Coordinate Transformation Software to provide seamless interoperability across multiple range coordinate systems**



# Foundation Initiative 2010 Mission



Currently, range systems tend to be **non-interoperable**,  
“**stove-pipe**” systems

The purpose of TENA is to provide the architecture and  
prototypes necessary to:

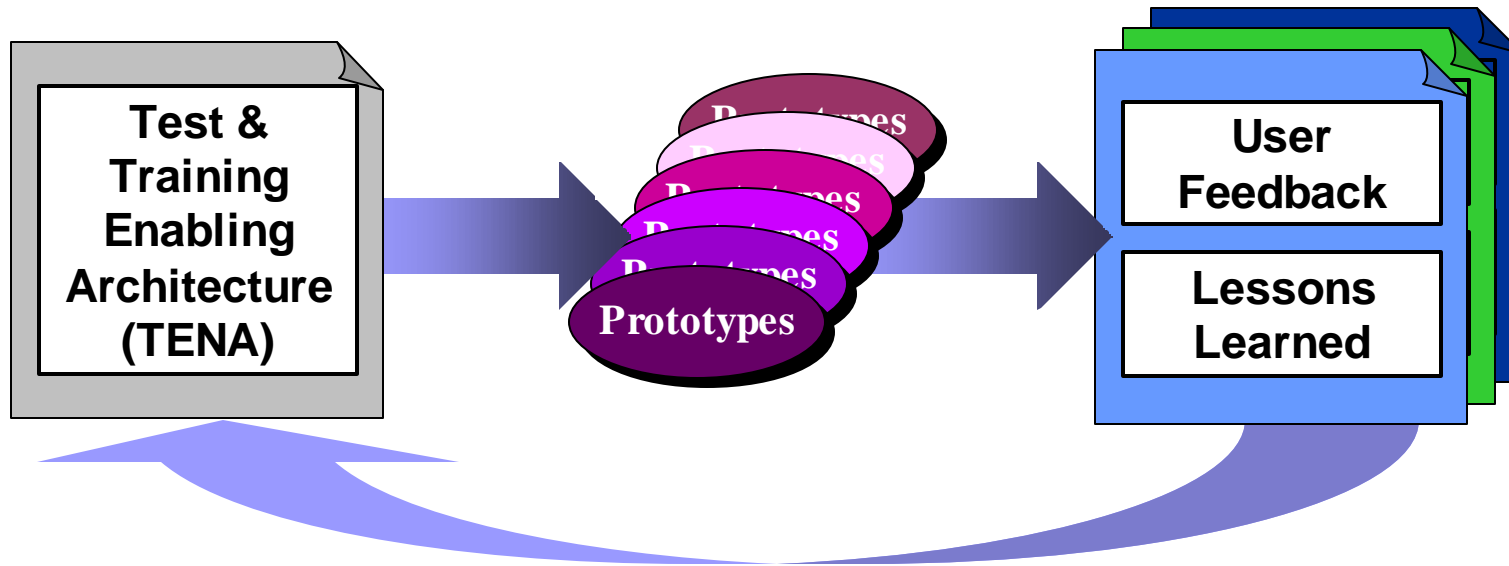
- Enable **Interoperability** among Ranges, Facilities, Simulations, C4ISR systems in a quick, cost-efficient manner
- Foster **Reuse** for Range asset utilization and for future developments

- ✍ Support the Warfighter (Joint Vision 2010/2020)
- ✍ Enable SBA, STEP, CEE, JSB, and JDEP
- ✍ Foster Test and Training Integration
- ✍ In the long term: SAVE MONEY!

*Lay the Foundation for Future Test and Training Range  
Instrumentation*



# Overall TENA Development Strategy



- TENA was revised based on user feedback and lessons learned from working software prototypes
- Current Version is **TENA 2002**
  - Architecture Document available for review and comment
    - <http://www.fi2010.org/documents/tena2002.pdf>
- TENA will be revised in the future based on future prototypes

***TENA is based on real-world tests at real ranges***



# Architecture Management Team (TENA AMT)



- **System Engineers & Technical Leads for the current major stakeholders of TENA**

- AAC, Eglin AFB FL
- NUWC, Newport RI
- RTTC, Huntsville AL
- PMRF Synthetic Range
- EPG, Fort Huachuca AZ
- WSMR, White Sands NM
- NAWC-AD, Pax River MD
- Virtual Proving Ground (VPG)
- Joint National Training Capability (JNTC)
- NAWC-WD, China Lake & Point Mugu CA
- Common Training Instrumentation Architecture (CTIA)
- National Unmanned Underwater Vehicle T&E Center (NUTEC)

***Meetings every  
4-8 weeks***

***Raytheon, Boeing,  
SAIC, APL, MIT LL,  
JITC, DMSO, NRL, &  
ATC also attend &  
participate***

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



# TENA's Technical Driving Requirements



- **Interoperability**

- The characteristic of a suite of independently-developed components, applications, or systems that implies that they can work together, as part of some business process, to achieve the goals defined by a user or users.

- **Reusability**

- The characteristic of a given component, application, or system that implies that it can be used in arrangements, configurations, or in enterprises beyond those for which it was originally designed.

- **Composability**

- The ability to rapidly assemble, initialize, test, and execute a system from members of a pool of reusable, interoperable elements.
- Composability can occur at any scale— reusable components can be combined to create an application, reusable applications can be combined to create a system, and reusable systems can be combined to create an enterprise.



# How is Interoperability Achieved?



- **Interoperability requires:**

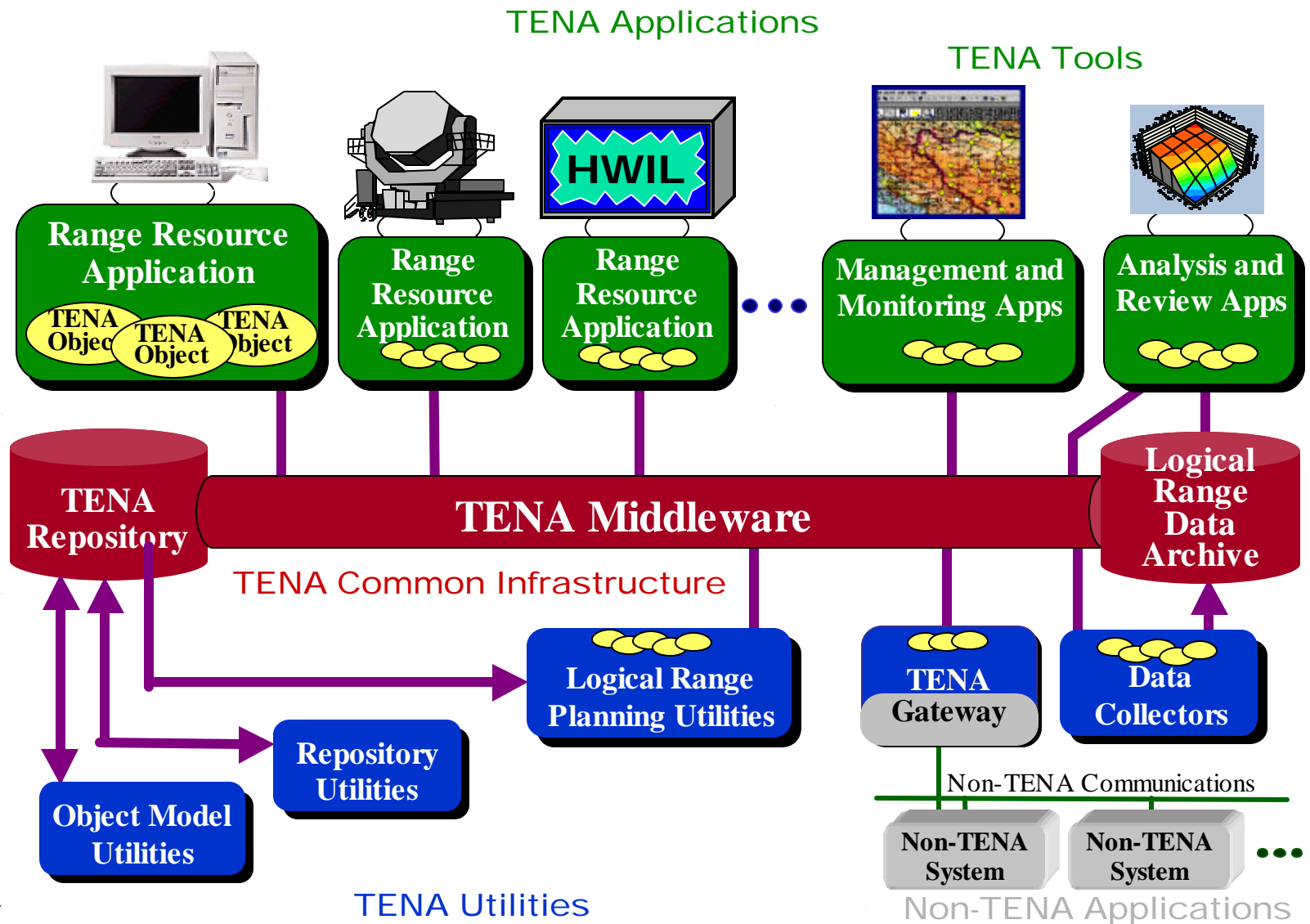
- A common architecture —————→ **TENA**
- An ability to meaningfully communicate
  - A common language —————→ **TENA Object Model (OM)**
  - A common communication mechanism —→ **TENA Middleware**
- A common context
  - A common understanding of the environment —————→ **TENA Object Model (Environment)**
  - A common understanding of time —————→ **TENA OM, Middleware**
  - A common technical process —————→ **TENA Technical Process**

- **Reuse and Composability require the above, plus**

- Well defined interfaces and functionality for the application to be reused —→ **Reusable Tools, Repository**



# TENA Architecture Overview







# TENA Uses the Concept of a Logical Range



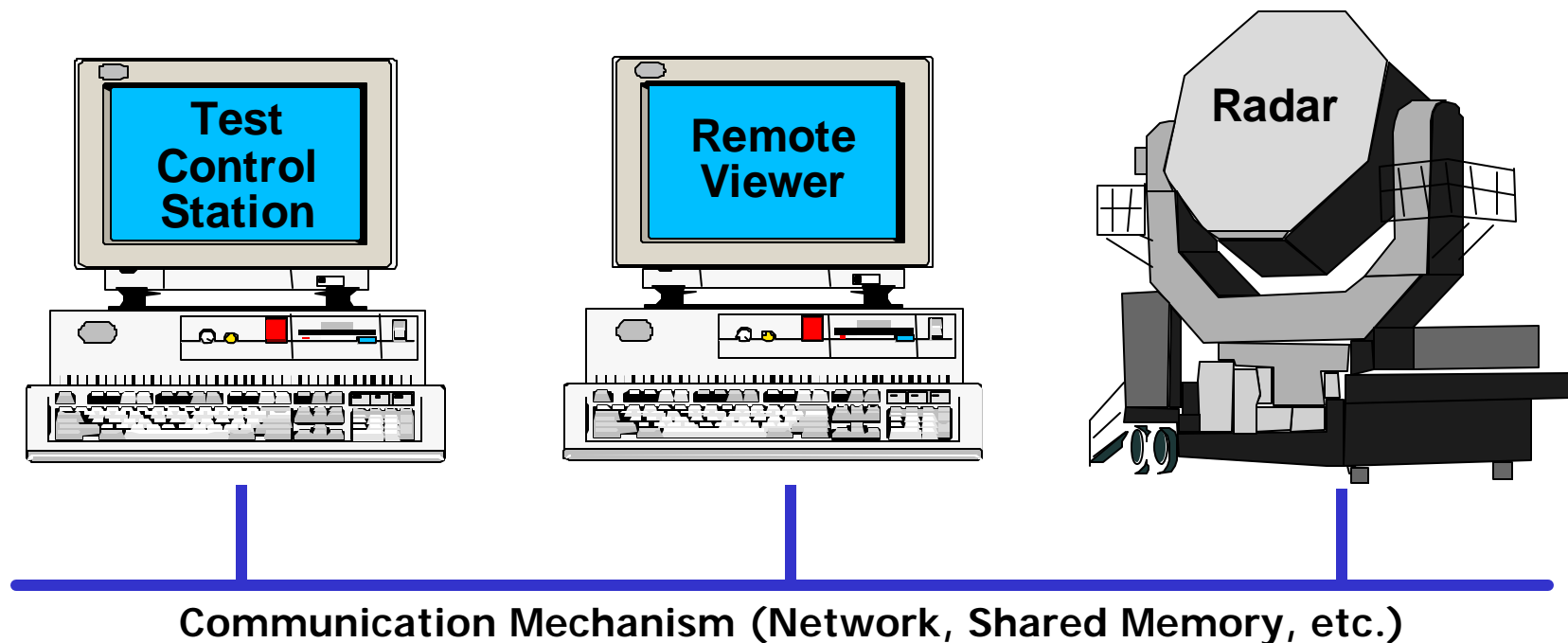
- **Logical Range** – a suite of **TENA Resources**, sharing a **common object model**, that work together for a given range event.
- **TENA Resources are:**
  - **Range Resource Applications** - compiled to use the services provided by the TENA Middleware for interaction,
  - **Gateway Applications** - to bridge TENA systems to legacy or other protocols or architectures, and
  - **TENA Tools and Utilities** - configured for a particular event.
- **Common Object Model**
  - **Logical Range Object Model (LROM)** – the object definitions used in a particular event.



# Logical Range Simple Example



**TENA specifies an architecture for range resources participating in **logical ranges****

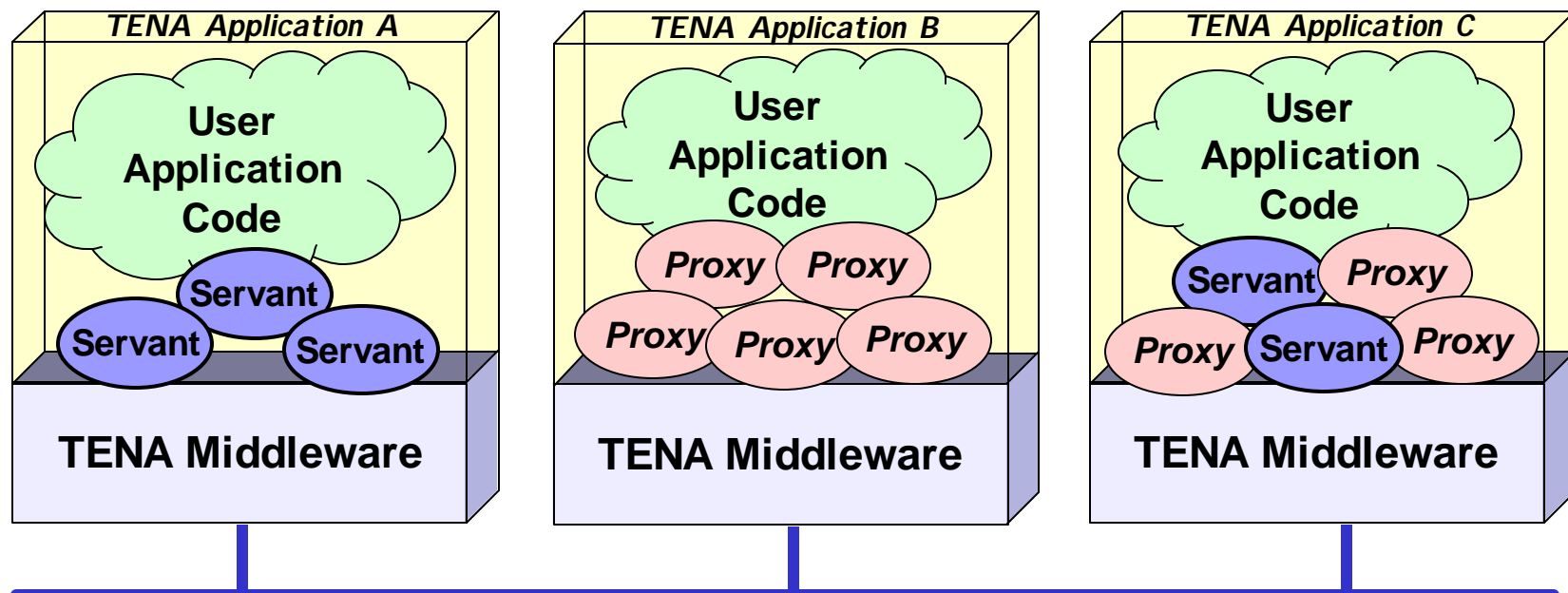




# Logical Range Simple Example



- TENA specifies a **peer-to-peer** architecture for logical ranges:
  - Applications can be both clients and servers simultaneously.
  - In their role as servers, applications serve TENA objects called “**servants**.”
  - In their role as clients, applications obtain “**proxies**,” representing other applications’ servants. Only servers can write to their servant objects’ publication state.
- The TENA Middleware, the TENA objects, and the user’s application code are compiled and linked together.



Communication Mechanism (Network, Shared Memory, etc.)



# Agenda



- TENA Overview and the Logical Range concept
- TENA Meta-Model – SDOs and Local Classes
- Using the SEDRIS SRM Coordinate Transformation Software to provide seamless interoperability across multiple range coordinate systems



# 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
  - No structs, no functions, no generics, no multiple inheritance
- **CORBA IDL**
  - Interfaces, structs, valuetypes, sequences, enumerations, multiple inheritance of interfaces, unions
  - No classes
- **HLA**
  - Classes (objects), interactions, attributes, single inheritance
  - No interfaces, no composition, no functions/methods, no ...



# Requirements for Defining the TENA Meta-Model



- Must support distributed computing
- Must be rich enough in features to support the object modeling needs of the entire test and training range community
  - Objects, Messages, Data Streams
- Must provide a semantic unification of information amenable to the creation of a simple, yet powerful, standard TENA Object Model
- Must be as easy to use and understand as possible given the above requirements

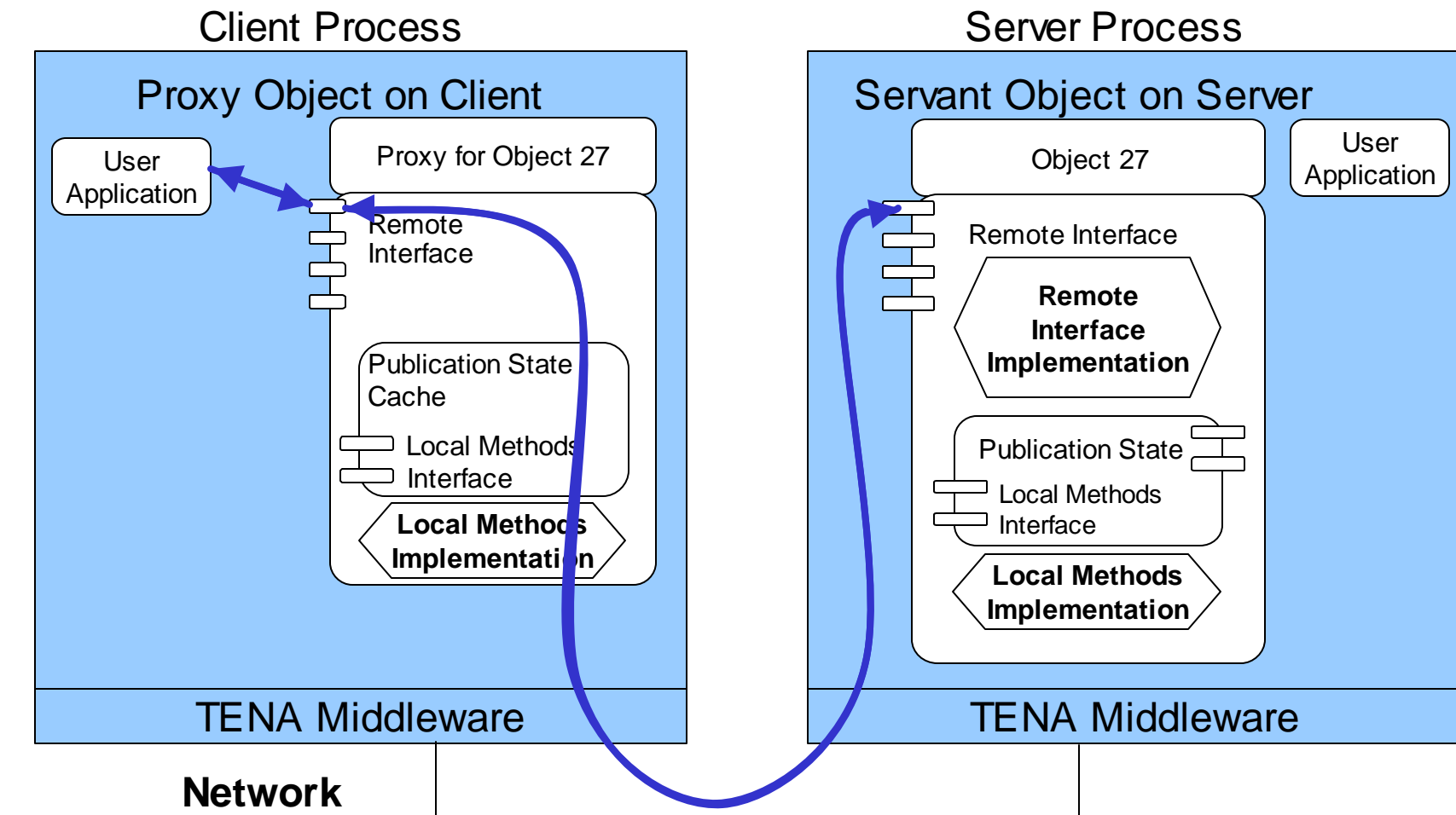
**These requirements led to the invention of the **Stateful Distributed Object**, combining the best features of CORBA and the HLA in one easy-to-use concept**



# Clients and Proxies, Servers and Servants



- Remote Method Invocation

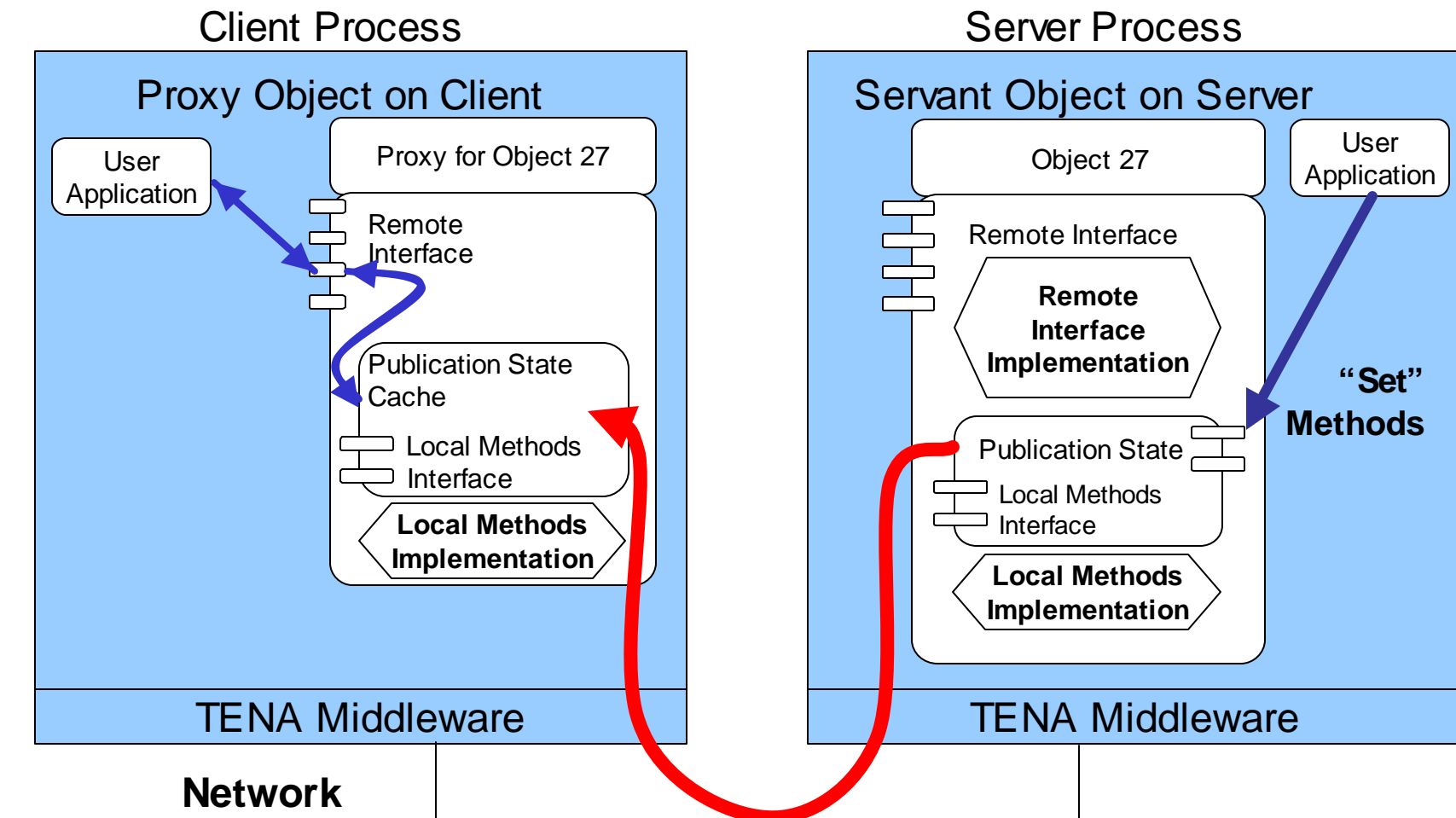




# Clients and Proxies, Servers and Servants



- Publication State Dissemination and Access



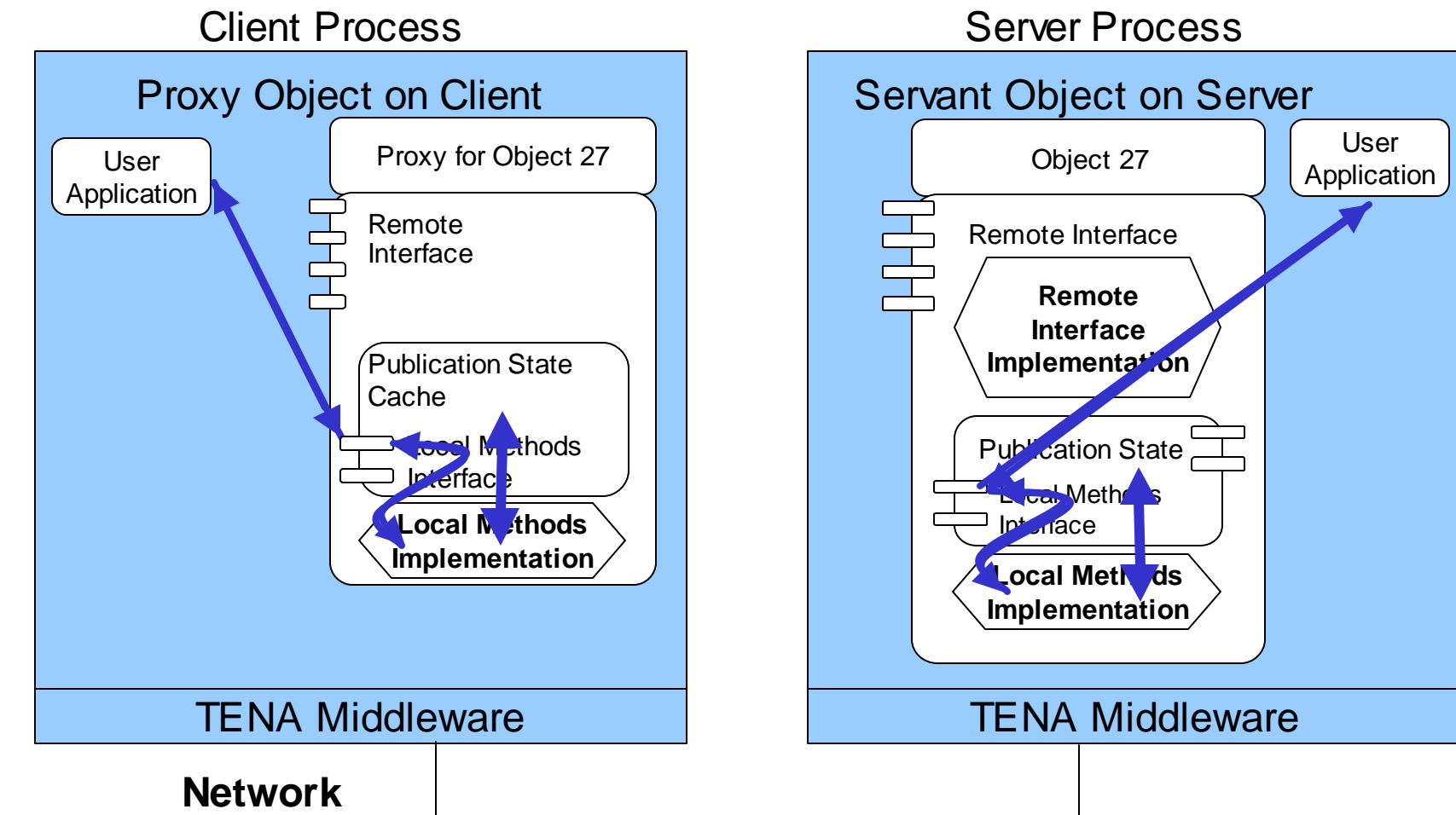




# Clients and Proxies, Servers and Servants



- Local Methods used on both Client and Server





# Agenda



- TENA Overview and the Logical Range concept
- TENA Meta-Model – SDOs and Local Classes
- Using the SEDRIS SRM Coordinate Transformation Software to provide seamless interoperability across multiple range coordinate systems



# How Do We Standardize a "Position" Object???



- How about:

```
class Position
{
    double x; // GCC in meters
    double y; // GCC in meters
    double z; // GCC in meters
}
```

- **Semantics are in the comments, not the code**
  - Both coordinate system and the units
- **Everyone is forced to use the same coordinate system**
  - Which one should we pick?
  - Why should everyone have to always perform coordinate conversions?
  - What about those people excluded because of our choice?
  - Do we really need to pick just a single coordinate system? Why?



# Principles for TENA TSPI OM Design



- **Encode semantics in the class definitions**
- **Allow users to choose their own coordinate system to work in**
  - Encode coordinate conversions inside the object model
  - Allow seamless interoperability between different coordinate systems
- **Support multiple coordinate systems simultaneously**
  - At first, support
    - Geocentric
    - Geodetic
    - Local Tangent Plane
    - Local Space Rectangular
    - Augmented Universal Transverse Mercator (A-UTM)
  - Eventually support additional coordinate systems as the community wishes, including, eventually, rotating coordinate systems
- **Support multiple coordinate system choices for position, velocity, acceleration, orientation, time, etc...**



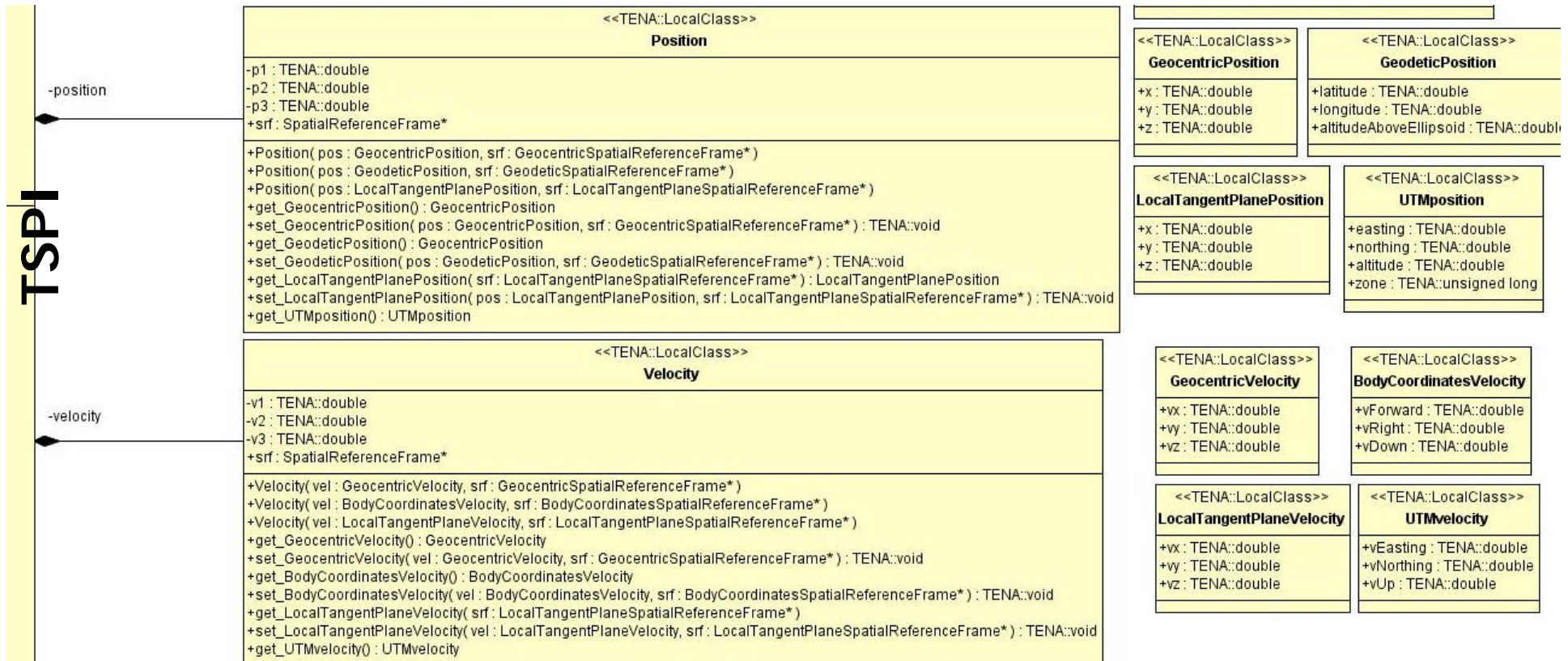
# Lazy Conversion for Higher Performance



- I use my coordinate system
- You use your coordinate system
- I send my information in my native format
- You receive my information in my format
- You only translate my information if you need to
- If we're using the same coordinate system
  - Great – no work and higher performance
- If we're not:
  - Still okay – translations performed on the fly at some cost
- **THIS IS REAL INTEROPERABILITY**
  - Everyone gets a choice
  - Some choices are better than others from a system perspective
  - But all choices work and work together
  - No one is forced into using coordinates they don't want and then doing mandatory translations all the time when they want to communicate with others!



# Draft (Simplified) TSPI Object Model



- Only Position and Velocity portion shown for clarity
- “Local Classes” are used to embed SEDRIS SRM-derived coordinate conversion functionality inside the objects themselves
- Each local class has a pointer to a reference frame object instance



# Summary



- **TENA provides an advanced interoperability architecture to the range community**
  - Many lessons were learned from other architectures, especially HLA
- **The TENA Meta-Model describes the rules for creating TENA objects**
  - Much more flexible and extensible than DIS, HLA, other architectures
- **TENA is also working toward a community-consensus-derived common object model**
- **TENA standard Time-Space-Position Information (TSPI) objects are going to be defined in accordance with the SEDRIS SRM and include:**
  - SRM coordinate conversions inside the objects defined as “local classes”
  - Support for multiple simultaneous interoperable coordinate systems