# An HLA based Man in loop Simulation framework for an Air Defense System

**By**
**Syed Rauf ul Hassan**

2005-NUST-MS PhD.CSE 05

Submitted to the Department of Computer Engineering
in fulfillment of the requirements for the degree of

Masters of Science
In
**Computer Software Engineering**

Thesis Supervisor
**Dr. Shoab Ahmad Khan**

College of Electrical and Mechanical Engineering
National University of Sciences and Technology
Rawalpindi,Pakistan
2009

# Acknowledgements

One of the great pleasures of writing a thesis is acknowledging the efforts of many people whose names may not appear on the cover, but whose hard work, cooperation, friendship, and understanding were crucial to the production of the thesis.

The report that you are holding is the result of many people's dedication. I am gratefully thankful to my supervisor Lt. Col. Dr. Shoab Ahmad Khan for encouraging my work at each step of my thesis and guiding me towards the right and realistic goals. Without his suggestions and guidance I would not been able to complete my thesis in efficient and timely manner.

I would like to thank Mr. Asad Waqar Malik and Mr. Anjum Latif for his support in the project. I would also like to thank Lt.Col Dr. Farooque Azam, Lt. Col. Dr. Rashid Ahmad and Dr. Aasia Khanum for being on my thesis committee.

This thesis would not have been possible without the love and support of my parents. They were my first teachers and inspired in me a love of learning and a natural curiosity.

*Dedicated to my parents, my wonderful wife, my brother and sisters who always encourage me for all the work I have done*

# Abstract

HLA is software architecture for creating computer models or simulation out of component models or simulations. At the core of the HLA is the software called ‘Run Time Infrastructure or RTI that is responsible for distribution and management of information. HLA has been used for the development of military simulation. In this work, we present a framework for an HLA based man-in-loop simulation. The main focus has been to simulate the real time command and control interface with the operator (man) interacting with the simulation through virtual console. Also the interfaces with sensors and weapons utilize the same interface and mechanism as used by actual sensors and weapons. This framework can be used for evaluation/training of the operator and testing of new weapons.

War games are widely used by the military to train troops and develop new ways of war fighting. There are various tools and softwares available that allow the user to create a synthetic/virtual battle field and test various scenarios. These includes OneSAF Testbed Baseline (OTB), STRIVE, MANA. OTB can interact with other live, virtual and constructive simulations based on HLA. In our work we present the ‘Air defense Simulation’ in which some components of the system are simulated/virtual whereas other components are live. In our simulation various simulated targets are generated by the operator and engaged either by the simulated or real gun. Sensors and weapons communicate with each other using standard messages. The simulation is tested in various scenarios for real time performance and detail analysis of results is presented.

# TABLE OF CONTENTS

# LIST OF TABLES

# LIST OF FIGURES

# List of Abbreviations

| | |
|---|---|
| ARPA | Advanced Research Project Agency |
| COTS | Commercial Of The Shelf |
| CGF | Computer Generated Forces |
| DCOM | Distributed Component Object Model |
| DIS | Distributive Interactive Simulation |
| DoD | Department of Defense |
| EADSIM | Enhanced Air Defense Simulation |
| FOM | Federation Object Model |
| HIL | Hardware in Loop |
| HLA | High Level Architecture |
| HMI | Human Machine Interface |
| LBTS | Lower Bound on Time Stamp |
| LRC | Local RTI Component |
| MIL | Man in Loop |
| MSMQ | Microsoft Messaging Queue |
| OMDT | Object Model Development Tool |
| RADAR | Radio Detection and Ranging |
| RPC | Remote Procedure Call |
| RTI | Run Time Infrastructure |
| SAF | Semi Automated Forces |
| SBA | Simulation Based Acquisition |
| SIMNET | **Sim**ulation **Net**working |
| STRICOM | **S**imulation, **T**raining and **I**nstrumentation **Com**mand |

# CHAPTER – 1

# INTRODUCTION

## 1.1. Introduction

Simulation is a representation of dynamics of a physical or abstract system. US DoD defines a simulation as “a method for implementing a model over time” and model as “a physical, mathematical, or otherwise logical representation of a system, entity, phenomenon, or process” [29]. Model and simulations have been extensively used for training, analysis, planning and demonstration of new technologies and system. The military uses simulator, simulations and exercises to emulate present or projected conditions.

Typical Simulation Scenario consists of scenario development, mission execution and data analysis stages. Scenario development stage involves the creation of various combat scenarios to be executed by a mission execution stage. Mission execution stage involves execution of scenario developed. It may include interaction with various hardware-in-loop and man-in-loop components depending upon the nature of simulation. Simulation also includes data analysis and logging facility. The potential use of this facility is to recreate the scenario and to access the operator performance.

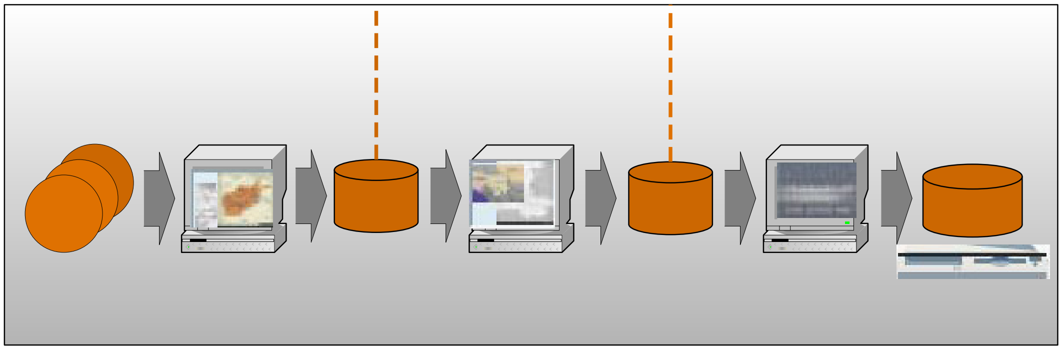


**Figure 1: Typical Simulation Scenario**

Simulation can be categorized into three types:-

1. Live Simulation
2. Virtual Simulation
3. Constructive Simulation

In live simulation real people use simulated equipment in real world. Human in loop or interactive simulations are live simulations. The most ancient and familiar type of simulation involving real soldiers, sailors and air crew operating real equipment are also categorized as live simulation. In virtual simulation real people uses simulated equipment in simulated world. In constructive simulation, simulated people use simulated equipment in simulated world. Example of constructive simulation includes exercises on the map and sand table.

Today Combat Simulation uses a variety of techniques from visualization to use of artificial intelligence, from data analysis generation tools to data analysis tools for improving the effectiveness of the simulation. With every passing day new techniques and methods are emerging in the field of electronic warfare. For example now a days concept of 'Computer Generated Forces' is emerging. They have the high potential in training, development and acquisition. They use the techniques from Artificial Intelligence to model and simulate military units ranging from large and complex vehicle system down to individual soldier.

In a combat simulation, it is important to have a real-time effect of the system behavior. Real Time Systems are those in which correctness of the system depends not only on the logical results of computation but also on the time at which the results are produced. Real-time systems are commonly divided into two categories: hard real-time systems and soft real-time systems. In hard systems, timing correctness is critically important and may not be sacrificed for other gains. In some cases, the timing correctness may be so important that criteria on logical correctness may be relaxed in favor of achieving timing correctness. In soft real-time system, time correctness is important but not critical. An occasional failure to observe deadline does not result in performance degradation. Soft real-time tasks are performed as fast as possible, but are not constrained by absolute deadlines, and their timing correctness may be sacrificed under special circumstances such as peak demands on the processor or the communication medium.

## 1.2. Hardware in Loop

Hardware-in-loop (HIL) simulation is the kind of simulation used in the development of complex real-time embedded systems. Software simulation of such systems is not possible because it does not run in real time with actual digital/analog signals. The testing on actual system is too much costly. So engineers resort to HIL simulation where real components interact with simulated components. This enhances productivity by reducing development cost. It also increases reliability and quality of the product.

Architecture of typical HIL system is depicted in the figure below:-

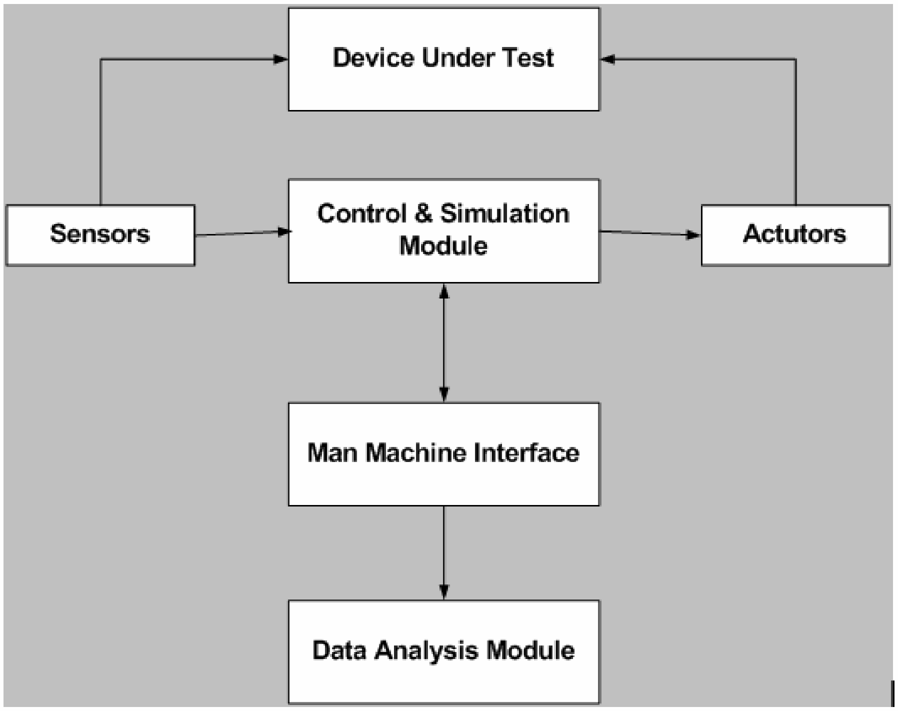


**Figure 2: Architecture of Typical HIL System**

A typical Hardware in Loop (HIL) system includes sensors to receive data from the control system, actuators to send data, a controller to process data, a human-machine interface (HMI) and a post simulation analysis module [25]. Companies like National Instrument have developed the product which facilitates engineer and scientist to develop HIL simulation.

## 1.3. Man in Loop

In man-in-loop simulation environment various decisions are made by actual decision makers operating within the simulated environment. One of the examples of the system is a remote missile controller. The system provides the operator at the base location with the real time image from the camera fitted on board. Operator observes the image and instructs the missile guidance system to make adjustment in flight path. Operator can also make other adjustment from the base location [1]. In Man in loop system operator/human must have a considerable amount of experience in using the system.

Most of the newly developed defense systems have the human-in-loop capability. These systems provide the capability to analyze and experiment with the existing and newly emerging strategic paradigms (like Network Centric Warfare or NCW). DARNOS (Dynamic Agents Representation of Networks of Systems) has human-in-loop capability [3].

# 1.4. Distributed Simulation Standards

Simulators developed prior to the 1980s were standalone system developed to carry out a specific task. These systems were quite expensive. However with the passage of a time a need arose to integrate individual simulations together by a network. SIMNET was the first such project developed by US Military.

## 1.4.1. Simulator Networking (SIMNET)

DARPA Simulator Networking project or SIMNET [27] was initiated by Jack A. Thorpe, with the help of Dr. Craig Fields in 1983, with a goal of developing high tech, realistic, networkable, microprocessor-based simulators that cost 100 times less than existing simulators. At the time SIMNET was developed, interactive simulation equipments were expensive stand-alone systems and it was not cost effective to replicate these facilities. SIMNET solved this problem by taking advantage of the technological advances in the field of computer network and computer graphics. Now individual simulation can be integrated together to form a network with individual simulator interacting in real-time.

Bolt, Beranek and Newman (BBN), Inc. and Perceptronics, Inc. with Delta Graphics, Inc. were the prime contractors that delivered the SIMNET development system. BBN developed software for vehicle simulation, networking, artillery resupply and semi automated forces. Delta Graphics developed the graphic system and terrain databases. Perceptronics built vehicle simulation shells, controls and sound systems.

The system itself consist of local and long-haul nodes of interactive simulators for command-and-control systems, tanks, fighting vehicles, artillery and fixed wing aircraft. Each node in the system is responsible for maintaining the state of simulation entities, with events and interactions communicated over the network. Event scheduling and conflict resolution is done in a distributed fashion. Every node creates its perception of the simulation individually based on what it receives from other nodes. In the time between each communication of simulation entity states each node executes micro-simulation of remote entities. This allows estimation of remote simulations to create a real-time interactive simulation on each node.

#### 1.4.1.1. Benefits and Drawback of SIMNET

The SIMNET protocol was developed at a time when the communication speed of networks outpaced the processing power of simulators. The result of this can be seen in the robust but often network-inefficient designs found throughout the protocol.

SIMNET consist of autonomous individual nodes. Therefore, the system is tolerant of single node failure, and non one node can bring the whole exercise down.

SIMNET uses “dead reckoning” mechanism to compensate for the network delays. The high tolerance for latency in SIMNET allows simulation to occur between nodes that are geographically separated.

The broadcast oriented PDU packets that SIMNET uses can get lost during transmission over the network, allowing some simulation nodes to receive updates while leaving out unreachable parts of the network.

### 1.4.2. Distributed Interactive Simulation (DIS)

In 1990, when ARPA transferred the SIMNET program to STRICOM, they changed the name to Distributed Interactive Simulation or DIS [28]. DIS emerged to become an open standard for distributed simulation, defined under IEEE Standard 1278. Simulation

Interoperability and Standards Organization (SISO), a sponsor committee for IEEE, works for improvement in the standard.

The system consists of number of computers interconnected together by a network. Each computer represents combat elements, defense elements, decision makers and logistic support elements. All of these participate in a simulated combat. The system is interactive in that the user of the system can influence the simulation. The user fights networked opposing forces which may be combination of virtual and semi automated forces.

### 1.4.3. High Level Architecture (HLA)

The High Level Architecture (HLA) is a standard framework that supports simulations composed of different simulation components. Traditional simulation lacks two properties reusability and interoperability. Reusability as the name indicates means that simulation components be used in other simulation scenarios and applications. Interoperability implies that individual simulation components on different distributed computer platform be combined together to work in real-time.

At the heart of HLA is the software called ‘Run Time Infrastructure’ (RTI) which is responsible for distribution of information and management of simulation units. The RTI provides the glue that unifies different concurrent simulation, known as ‘federation execution’. The modeling and implementation of federates (or individual simulation components) can be carried out independently of RTI, as a layer of abstraction exists between federates and the data distribution mechanism of HLA.

RTI provides services which can be grouped into six categories:-

1. Federation Management
2. Declaration Management
3. Object Management
4. Ownership Management
5. Data Distribution Management
6. Time Management

The application of HLA extends beyond military simulation system, as many industrial and commercial organizations have now adopted it for their modeling and simulation systems. The importance of HLA to defense simulation and modeling community has been highlighted by the steps taken to adopt it as unifying standard for all simulation and modeling systems.

HLA is defined under IEEE standard 1516. Before the development of IEEE standard, HLA was sponsored by US Defense modeling and simulation office.

#### 1.4.3.1. Benefits and Drawback of HLA

As a software architecture standard that is independent of any particular implementation HLA is currently unchallenged. It arose from the evolution of previous standards such as DIS to meet the emerging requirements of the US defense community. The standard leaves out implementation details in favor of more general rules and guidelines upon which to base the implementation. Thus the combination of features implemented in the RTI will be based on the HLA specification, but many vary according to the needs of the system to be developed.

The rules and guidelines specified in the HLA allows for the creation of readily expandable synthetic environments in a cost effective manner. HLA allows the addition of new federates to the system without a major overhaul of existing technology. Once the simulations models have been developed they are highly reusable as a federate from one HLA implementation can be configured to run as a part of another. Being able to interoperate various simulation systems and re-use components help to economize the acquisition of simulation systems in the long run.

## 1.5. Summary

In the modern era, it is very important for the defense forces to have a proper training for peace and war operations. These can be learned through the variety of simulation tools. Especially, man-in-loop simulation has significant importance in training operator for efficient use of the system.

# CHAPTER-2

## LITERATURE SURVEY

## 2.1. Introduction

Simulation is an imitation of some real thing, state of affairs, or process. The act of simulating something generally entails representing certain key characteristics or behavior of a selected physical or abstract system. Simulation has found its application in the modeling of natural systems, performance optimization, safety engineering, testing, verification, training and education.

There are numerous classes of simulation. Physical and interactive simulation is one of them. In Physical simulation, physical objects are substituted for real things. Man-in-loop simulation is a special kind of physical simulation in which operator is involved, such as flight simulator or a driving simulator. Human in the loop simulation can include a computer simulation as a so-called synthetic environment.

Man in the loop simulation has found its application in military simulation or war games. Military simulations are models in which theories of warfare can be tested and refined without the need for actual hostilities.

## 2.2. Air Defense Simulation

Air Warfare is the most rapid, intense, and devastating type of warfare. Due to the fast pace, uncertain, and dangerous aspects of air warfare, the air force man must be trained extensively in the fundamental tenets of these operations in order to effectively protect the aircraft, high-value units, and other military assets. Simulation is one of the many ways of training.

There has been a lot of work done in the last ten years on the development of simulation system. The Air-Defense Commander (ADC) is one of them. ADC is a top-view, dynamic, Java language-based, graphics-driven software implementation of an AEGIS Cruiser Combat Information Center (CIC) team performing the Battle Group Air-Defense Commander duties in the Arabian Gulf region [12]. The objective of the system was to evaluate the performance of ADC. It allows the operator to configure a wide variety of simulation parameters to create unique and realistic air scenario. Also the operator can modify the scenario “on-the-fly” to explore different potential outcomes. All the events in the scenario are logged for reconstruction of particular scenario.

The Area Air-Defense Commander (AADC) Battle Management System was designed and developed by John Hopkins University’s Advanced Physics Laboratory (APL) for the United States Navy. Major objective for the development of AADC was theatre-wide, strategic and operational planning by the AADC. The AADC provides a single, integrated picture of the battle-space so that a joint commander can quickly gather data on air and missile attacks and defend against them. Also, the AADC System would allow the air-defense staff to rapidly create, modify, and evaluate plans through system’s automated uses which substantially reduced the time of the process. The system was developed after Gulf War. Prior to the development of system air-defense planning was done manually by 10-15 peoples and normally would take hours.

The Tactical Decision-Making under Stress (TADMUS) study was one of the first comprehensive explorations into the causes of the USS Vincennes incident (in which an Iranian civilian aircraft was shot down). The TADMUS study evaluates the system from the Human Computer Interaction perspective and found major flaws which resulted in the degradation of the operator performance. This system ultimately resulted in the development of Decision Support System (DSS) with improved consoles. The improved consoles display data in graphical format and facilitate decision making process by the operator under stressful conditions.

The Multi-Modal Watch Station (MMWS) program was a four-year project focused on the development of specialized watch station consoles that incorporated improved human-computer interface (HCI) designs to improve the performance of watch-teams during battle group air defense and land-attack warfare operations [13]. The primary focus of the system was to develop the system that incorporates the user requirement and task instead of forcing the operator to adapt to the system.

Air Threat Assessment studies provide a theoretical and applied basis by defining specific cue-data relationships and detailing the cognitive process involved in air-defense assessment [14]. The process was incorporated into a model which was validated by the air-defense decision makers.

There have been numerous games that attempt to model the military operation and planning. These include games like ‘Unreal Tournament’, ‘Quake’, ’Medal of Honor’, ‘Harpoon’,’ Strike Fleet’ and ’Fifth Fleet’. These games have become template for many military research projects. However these games are different from simulation, because the overall objective of Air Defense Simulation is to get insight into the performance of battle group air defense.

QualNet® has developed high speed network models that produce realistic simulation of network centric warfare.

## 2.3. High Level Architecture

The High Level Architecture (HLA) was developed by the Defense Modeling and Simulation Office (DMSO) of the Department of Defense to meet the increasing need of defense related projects [15]. The main aim and objective of the HLA was to provide highly reusable simulation components that support distributed simulation. It must be noted that HLA is architecture; it does not provide implementation details.

There are numerous battle simulators that use simulation infrastructure provided by HLA. JSAF is one of the simulators that provide computer generated forces (CGF) such as land vehicles, aircraft and ships in dynamic battle scenarios, and operates in a Linux environment (Fedora Core 3). All federates in this simulation use a commercial runtime infrastructure (RTI) by MäK. The latest version of CAE's simulator, called STRIVE 2.0, is a Microsoft Windows application and provides its own full CGF capability, running under the Defense Modeling and Simulation Office (DMSO) RTI.

One of the requirements of man-in-loop simulation is that simulation must be able to work in real time. So Run Time Infrastructure (RTI) must provide services to federate and federation within the bounded response time and also behave in a predictable fashion. There have been numerous protocols proposed for improving the RTI communication performance. Virtual reality transfer Protocol (VRTP) is proposed by

Bruzman et. al. [17] that support the real time interaction between federates. However there is a cost associated with the implementation of such customized protocol and they are not suitable for all applications. Also, numerous researchers have proposed the use of Quality of Service (QoS) and real time operating system (RTOS) for an RTI to provide services in real time.

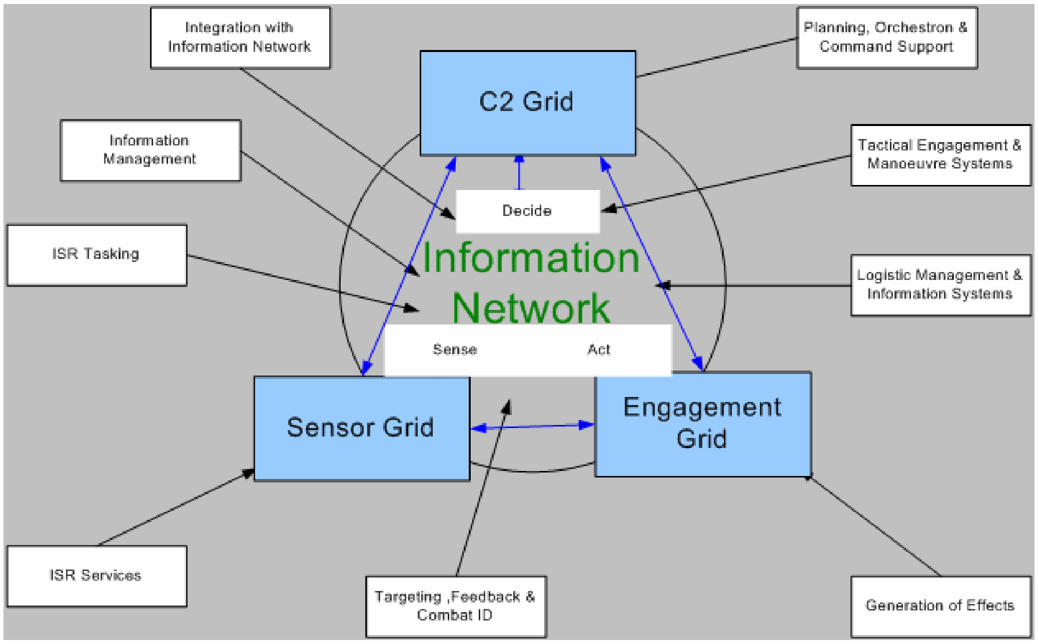
Most of the techniques proposed by researcher fall into the six main categories: Network QoS [18,19], RTI multi-threaded asynchronous process [20], Preemptive priority scheduling [21], globally scheduling service, Real time optimized RTI services, and special purpose transmission protocol . Azeedine et. al. [16] has proposed a novel approach to real-time RTI based distributed simulation system. They have proposed an optimized data distribution management (DDM) scheme for filtering out the irrelevant data exchanged among the simulated entities. Also, they have used the modified the time management (TM) LBTS value calculation algorithm. These modifications into the RTI ensure real time performance.

## 2.4. Man in Loop Simulation

There are quite a few applications that have man-in-loop support. Air Defense System Simulation Framework (ADSSF) has the capability of "perception visibility" or man-in-loop reactivity. It has the DIS, HLA and other interfaces and augments legacy simulations (SUPPRESSOR, EADSIM, and JIMM).

QualNet® also support Hardware in the loop and man in the loop simulation.

The Dynamic Agents Representation of Networks of Systems (DARNOS) provides defense community with a modeling and simulation capability for Network Centric Warfare (NCW) analysis. DARNOS was designed for constructive simulation. The simulation infrastructure used by DARNOS was the one provided by BattleModel. In 2005 DARNOS was extended with Human-In-Loop (HIL) capability in support of Headway 2005. DARNOS provides a unique capability for analysts and commanders to explore different options available for structuring Command and Control system. The HIL capability of DARNOS can be used to support analysis of different information dissemination paradigms, and experiment with different networking, command and control architectures [22].



Integration with Information Network
Planning, Orchestron & Command Support
C2 Grid
Information Management
Tactical Engagement & Manoeuvre Systems
Decide
ISR Tasking
Information Network
Logistic Management & Information Systems
Sense
Act
Sensor Grid
Engagement Grid
ISR Services
Targeting ,Feedback & Combat ID
Generation of Effects

**Figure 3: A System for Network Centric Warfare Analysis**

There are already inventions available relating to the control of remotely controlled missile by human-being [23].The missile is remotely guided on its flight toward target or it accepts update to the pre-planned target from a person/operator at the base location. Such system requires the operator who has a considerable amount of experience in remotely "flying" the missile, gained through simulators or live exercises, and who must be adept at interpreting the video imagery and evaluating the missile capability of prosecuting the correct target in real time.

## 2.5. Network Centric Warfare

Network Centric Warfare is a relatively new term and traces its origins to 1996 when Adm. William Owens introduced the concept of a “systems of systems” in a paper of the same name published by the Institute National Security Studies. The US DoD has mandated that the Global Information Grid (GIG) will be primary technical framework to support NCW/NCO under which all advanced weapon platform, sensor systems and command and control centers be linked together.

## 2.6. Summary

This chapter provided the overview of the trend in the simulation with special emphasis on Air Defense Simulation.

# CHAPTER –3

# SYSTEM DESCRIPTION AND DESIGN

## 3.1. Introduction

The Air defense architecture compromises of several battalion command post's (BCP), which are installed throughout the border areas and monitor any suspicious moment. BCP consist of radar, sensor and execution center whose primary functionality is to send the detected information to the next higher level and receive the instruction from higher level and execute it with the help of battalion etc. Above the BCP there are Regiment Command Post's (RCP) which are fewer in number as compared to BCP's. Usually several BCP's are monitored by a single RCP, depend upon the complexity of region. The primary function of RCP is to receive the information from BCP and send it to next higher level. Thus its make the information flow possible. RCP is also connected to some other RCP; the information passed on during simulation is very sensitive and must be processed with in a limited time, so that RCP always have more than one path for information to send next higher level. The next stage consist of forwarding router, they are mobile nodes, which can change its position during execution. The final stage in Joint Air Defense Command (JADC) takes the information from different RCP's, makes a decision and sends the instruction back to RCP for appropriate action. JADC are much fewer in number as compared to RCP's. JADC's are also connected with other JADC.

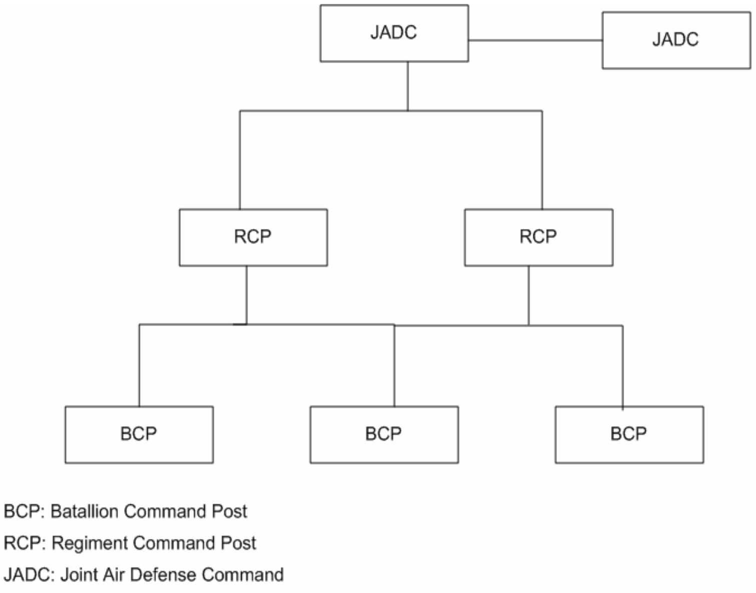

JADC
JADC
RCP
RCP
BCP
BCP
BCP
BCP: Batallion Command Post
RCP: Regiment Command Post
JADC: Joint Air Defense Command

**Figure 4: Air Defense System Architecture**

HLA simulation is made of number of HLA federates and are called federation. There can be multiple instances of a particular type of federate, for example several Boeing 747 simulations or F-16 simulations, in a given federation, and this number can change as the simulation continues.

Our ‘Air Defense Simulation’ consists of following types of federates:-

1. Battalion Command Post’s (BCP)
    a. Sensors (e.g RADARs)
    b. Guns
2. Regiment Command Post’s (RCP)
3. Joint Air Defense Command’s (JADC)
4. Targets (e.g Fighter Jets)

User can create BCPs and Targets on the digitized map of the Pakistan. For RADAR and Guns user can specify their type and ranges. For targets, trajectory can be specified.

The system consists of numerous other components besides RADAR, gun and target federate. There is a federate which allows operator to create a scenario on a digitized map. It displays the real time target location as the simulation progress. Operator designates the target on the map and then issue guidance command to the gun. Guidance commands can use electrical signal, an electromagnetic signals, light transmitted through an optical fiber, or satellite link relay. ‘UK Tactical Data System Reference Guide’ specifies a list of the mechanism currently used for command and control by the strategic system [2].

Architecture of our MIL simulation is depicted below:-

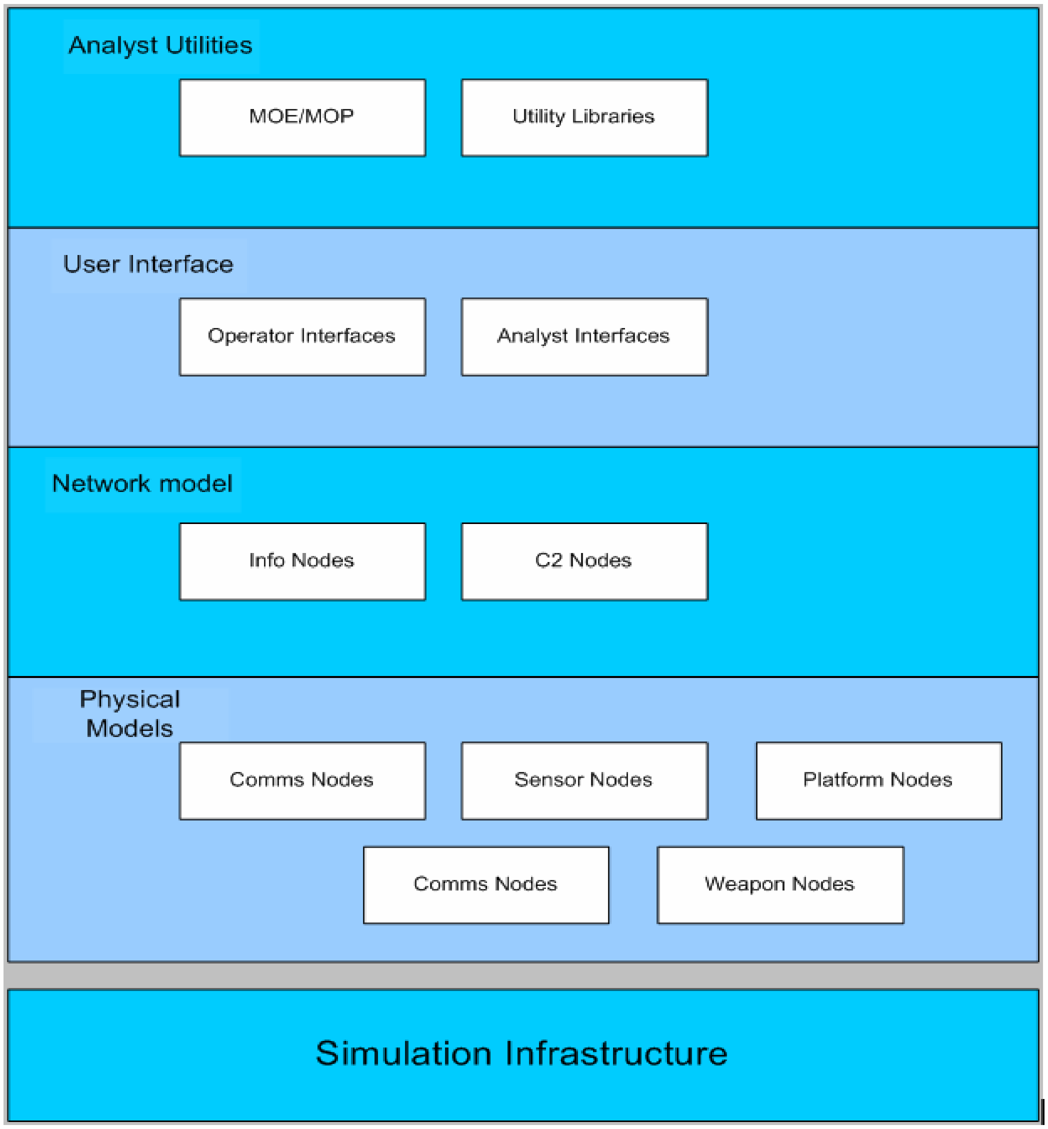


**Figure 5: Functional Architecture of MIL Simulation**

The physical components include sensors nodes, communication node, platform nodes and weapon nodes. These components model the physical characteristics of the nodes that exist within a scenario. The network component models the command and control, and information sharing responsibilities of the nodes. The User Interface component includes the Analyst Interface and Operator Interface. The Operator interface module is used to support the experimental MIL activity. The Analyst Utility includes the Measures of Performance (MOP) and Measures of Effectiveness (MOE) component used to evaluate the adopted organizational structures. Some measures are collected and displayed during the experiment on the Analyst Interfaces [4]

In our present scenario we implement each of the nodes as a federate in a federation. Also one of the nodes is referred to as super node which acts as a scenario generator. It generates various nodes in the system.

To support interfacing with actual weapons, we add a virtual federate before the actual weapon. Virtual federate act as a gateway between the simulated weapon and actual weapon.

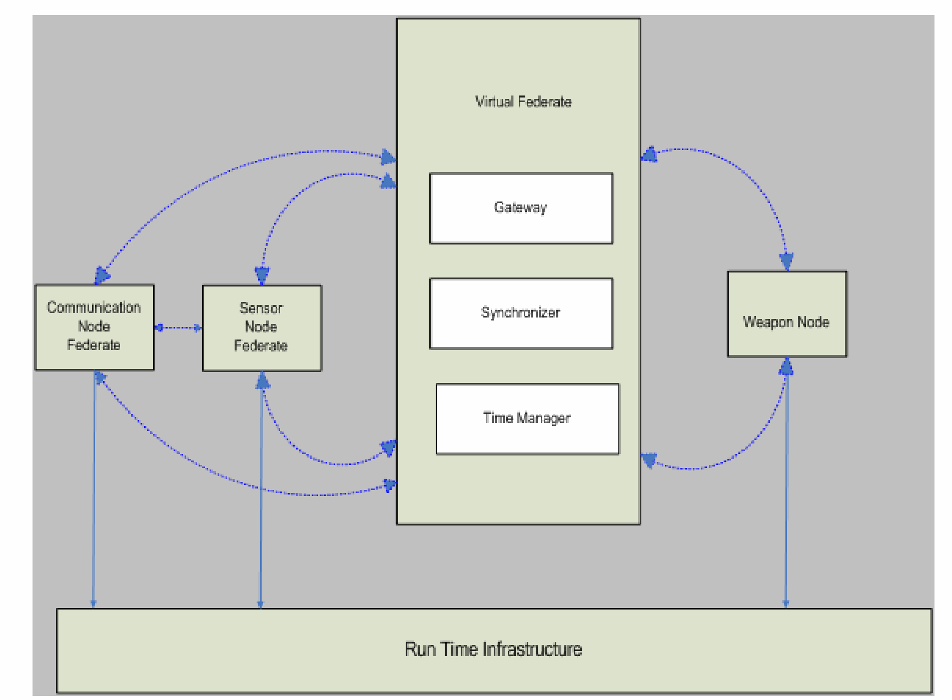


**Figure 6: Air Defense Simulation Framework**

## 3.2. HLA

HLA consist of three components:-

1. HLA Rules
2. Interface Specification

3. Object Model Template (OMT)

### 3.2.1. HLA Rules

Federation Rules ensure proper interaction of simulation in federation. They describe the responsibility of simulation and federate. HLA compliant simulation must follow these rules. HLA rules are divided into two groups each consist of five rules. Of these five rules are for HLA federation and five are for HLA federate.

### 3.2.2. HLA Interface Specification

The Interface specification defines standard for Run-Time Infrastructure. Interface specification provides description of the functionality of each service and requires arguments and pre-conditions necessary for the use of the service. It also contains information about the related services. Interface specification consists of following type of information:-

1. Interface name and description of service
2. Supplied arguments
3. Returned arguments
4. Pre Conditions
5. Post Conditions
6. Exceptions
7. Related Services

RTI (Run Time Infrastructure) is the software that implements HLA interface specification and provides common services to simulation system. RTI separates simulation and communication. RTI software comprises of:-

1. RTI Executive Process (RtiExec)
2. Federation Executive Process (FedExec)
3. LibRTI library

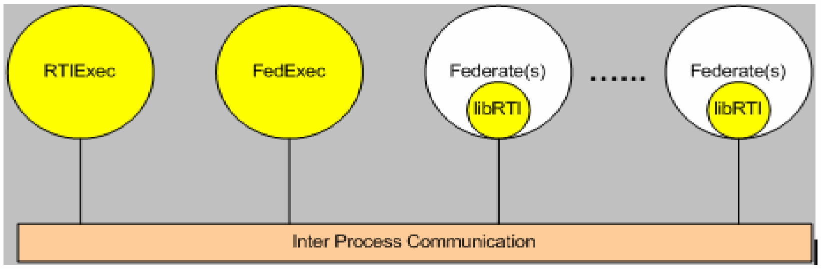


**Figure 7: RTI Components**

RTIExec is a globally known process. Its purpose is the creation and destruction of fedExecs. FedExec allows federates to join and resign, and facilitate data exchange between participant federates. LibRTI provides services such as federation management, data management, object management and time management to the participating federates. These services are provided by the routines in the class 'RTIAmbassador'. In order to create a new federate user needs to provide the implementation for the abstract class 'FederateAmbassador'.

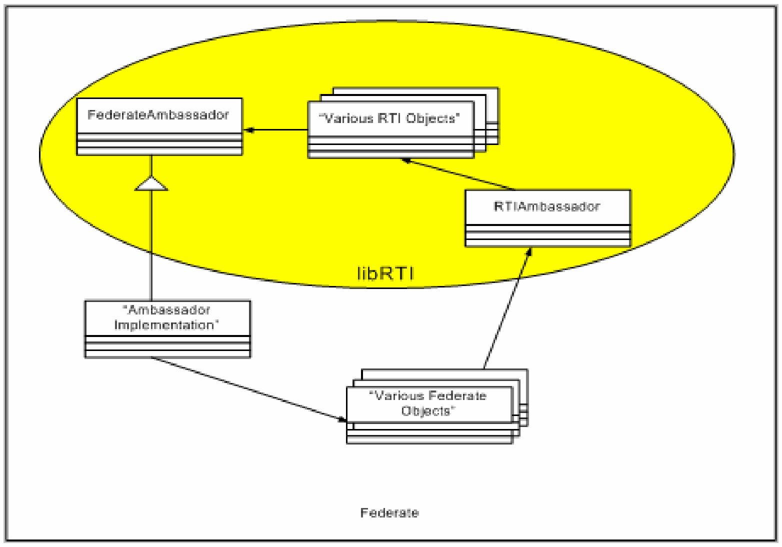


**Figure 8: RTI and Federate Responsibilities**

Each Federate also maintains two queues for receiving data from other federates. These are:-

1. FIFO Receive Queue

2. Priority Time Stamp Queue

Information between federates is exchanged through RTI. The RTI provides functions for synchronizing activities between federates participating in a federation. It is also possible to specify explicit synchronization points.

### 3.2.3. HLA Object Model Template

HLA requires that federations and individual federates be described by an object model which identifies the data exchanged at runtime in order to achieve federation objectives. The primary purpose of HLA object model is to facilitate reusability and interoperability.

HLA object model consist of two sub models: HLA simulation object model (SOM) and HLA federation object model (FOM). HLA simulation object model (SOM) is used to specify the capability of an individual federate in relation to the whole federation. HLA federation object model (FOM) specifies information about data exchanged among federates. It includes enumeration of all objects and interaction classes, along with attributes and parameters that characterize these classes.

All the information about HLA FOM and SOM is specified in the "Federation Execution Data (FED) File". This file also contains information required by the RTI for the execution of the simulation. Below are the excerpts from the FED file used by Air Defense Simulation:-

```
(FED
  (Federation SimulatePopulation)  ;; we choose this tag
  (FEDversion v1.3)          ;; required; specifies RTI spec version
(spaces
  (space growth
    (dimension population
    )
    (dimension rate
    )
  )
)
  (objects
    (class ObjectRoot        ;; required
       (attribute privilegeToDeleteObject reliable timestamp);; realiable mean
       tcp, best effort udp
      (class RTIprivate);; necessary
      (class Country
        (attribute name reliable timestamp growth);; growth mean region
        (attribute population reliable timestamp growth)
      )
      (class Manager         ;; Manager class and subclasses are required
         (class Federation
           (attribute FederationName reliable receive)
           (attribute FederatesInFederation reliable receive)
           (attribute RTIversion reliable receive)
           (attribute FEDid reliable receive)
           (attribute LastSaveName reliable receive)
           (attribute LastSaveTime reliable receive)
           (attribute NextSaveName reliable receive)
           (attribute NextSaveTime reliable receive))
         (class Federate
           (attribute FederateHandle reliable receive)
           (attribute FederateType reliable receive)
           (attribute FederateHost reliable receive)
           (attribute RTIversion reliable receive)
           (attribute FEDid reliable receive)
           (attribute TimeConstrained reliable receive)
           (attribute TimeRegulating reliable receive)
           (attribute AsynchronousDelivery reliable receive)
           (attribute FederateState reliable receive)
           (attribute TimeManagerState reliable receive)
           (attribute FederateTime reliable receive)
           (attribute Lookahead reliable receive)
           (attribute LBTS reliable receive)
           (attribute MinNextEventTime reliable receive)
           (attribute ROlength reliable receive)
           (attribute TSOlength reliable receive)
           (attribute ReflectionsReceived reliable receive)
           (attribute UpdatesSent reliable receive)
           (attribute InteractionsReceived reliable receive)
           (attribute InteractionsSent reliable receive)
           (attribute ObjectsOwned reliable receive)
           (attribute ObjectsUpdated reliable receive)
           (attribute ObjectsReflected reliable receive)))
    )                        ;; end ObjectRoot
  )                          ;; end objects
  (interactions
    (class InteractionRoot reliable timestamp
      (class TransferAccepted reliable timestamp
         (parameter servingName)
      )
      (class RTIprivate reliable timestamp)
      (class Manager reliable receive
          (class SimulationEnds reliable receive)
          (class Federate reliable receive
             (parameter Federate)
             (class Request reliable receive
               (class RequestPublications reliable receive)
               (class RequestSubscriptions reliable receive)
               (class RequestObjectsOwned reliable receive)
               (class RequestObjectsUpdated reliable receive)
               (class RequestObjectsReflected reliable receive)
               (class RequestUpdatesSent reliable receive)
               (class RequestInteractionsSent reliable receive)
               (class RequestReflectionsReceived reliable receive)
               (class RequestInteractionsReceived reliable receive)
               (class RequestObjectInformation reliable receive
                 (parameter ObjectInstance)))
             (class Report reliable receive
               (class ReportObjectPublication reliable receive
                 (parameter NumberOfClasses)
                 (parameter ObjectClass)
                 (parameter AttributeList))
               (class ReportObjectSubscription reliable receive
      (parameter NumberOfClasses)
      (parameter ObjectClass)
      (parameter Active)
      (parameter AttributeList))
    (class ReportInteractionPublication reliable receive
      (parameter InteractionClassList))
    (class ReportInteractionSubscription reliable receive
      (parameter InteractionClassList))
    (class ReportObjectsOwned reliable receive
      (parameter ObjectCounts))
    (class ReportObjectsUpdated reliable receive
      (parameter ObjectCounts))
    (class ReportObjectsReflected reliable receive
      (parameter ObjectCounts))
    (class ReportUpdatesSent reliable receive
      (parameter TransportationType)
      (parameter UpdateCounts))
    (class ReportReflectionsReceived reliable receive
      (parameter TransportationType)
      (parameter ReflectCounts))
    (class ReportInteractionsSent reliable receive
      (parameter TransportationType)
      (parameter InteractionCounts))
    (class ReportInteractionsReceived reliable receive
      (parameter TransportationType)
      (parameter InteractionCounts))
    (class ReportObjectInformation reliable receive
      (parameter ObjectInstance)
      (parameter OwnedAttributeList)
      (parameter RegisteredClass)
      (parameter KnownClass))
    (class Alert reliable receive
      (parameter AlertSeverity)
      (parameter AlertDescription)
      (parameter AlertID))
    (class ReportServiceInvocation reliable receive
      (parameter Service)
      (parameter Initiator)
      (parameter SuccessIndicator)
      (parameter SuppliedArgument1)
      (parameter SuppliedArgument2)
      (parameter SuppliedArgument3)
      (parameter SuppliedArgument4)
      (parameter SuppliedArgument5)
      (parameter ReturnedArgument)
      (parameter ExceptionDescription)
      (parameter ExceptionID)))
  (class Adjust reliable receive
    (class SetTiming reliable receive
      (parameter ReportPeriod))
    (class ModifyAttributeState reliable receive
      (parameter ObjectInstance)
      (parameter Attribute)
      (parameter AttributeState))
    (class SetServiceReporting reliable receive
      (parameter ReportingState))
    (class SetExceptionLogging reliable receive
      (parameter LoggingState)))
  (class Service reliable receive
    (class ResignFederationExecution reliable receive
      (parameter ResignAction))
    (class SynchronizationPointAchieved reliable receive
      (parameter Label))
    (class FederateSaveBegun reliable receive)
    (class FederateSaveComplete reliable receive
      (parameter SuccessIndicator))
    (class FederateRestoreComplete reliable receive
      (parameter SuccessIndicator))
    (class PublishObjectClass reliable receive
      (parameter ObjectClass)
      (parameter AttributeList))
    (class UnpublishObjectClass reliable receive
      (parameter ObjectClass))
    (class PublishInteractionClass reliable receive
      (parameter InteractionClass))
    (class UnpublishInteractionClass reliable receive
      (parameter InteractionClass))
    (class SubscribeObjectClassAttributes reliable receive
      (parameter ObjectClass)
      (parameter AttributeList)
      (parameter Active))
    (class UnsubscribeObjectClass reliable receive
      (parameter ObjectClass))
    (class SubscribeInteractionClass reliable receive
      (parameter InteractionClass)
      (parameter Active))
    (class UnsubscribeInteractionClass reliable receive
      (parameter InteractionClass))
    (class DeleteObjectInstance reliable receive
      (parameter ObjectInstance)
      (parameter Tag)
      (parameter FederationTime))
    (class LocalDeleteObjectInstance reliable receive
      (parameter ObjectInstance))
    (class ChangeAttributeTransportationType reliable receive
      (parameter ObjectInstance)
      (parameter AttributeList)
      (parameter TransportationType))
    (class ChangeAttributeOrderType reliable receive
      (parameter ObjectInstance)
      (parameter AttributeList)
      (parameter OrderingType))
    (class ChangeInteractionTransportationType reliable receive
      (parameter InteractionClass)
      (parameter TransportationType))
    (class ChangeInteractionOrderType reliable receive
      (parameter InteractionClass)
      (parameter OrderingType))
    (class UnconditionalAttributeOwnershipDivestiture reliable
  receive
      (parameter ObjectInstance)
      (parameter AttributeList))
    (class EnableTimeRegulation reliable receive
      (parameter FederationTime)
      (parameter Lookahead))
    (class DisableTimeRegulation reliable receive)
    (class EnableTimeConstrained reliable receive)
    (class DisableTimeConstrained reliable receive)
    (class EnableAsynchronousDelivery reliable receive)
    (class DisableAsynchronousDelivery reliable receive)
    (class ModifyLookahead reliable receive
      (parameter Lookahead))
    (class TimeAdvanceRequest reliable receive
      (parameter FederationTime))
    (class TimeAdvanceRequestAvailable reliable receive
      (parameter FederationTime))
    (class NextEventRequest reliable receive
      (parameter FederationTime))
    (class NextEventRequestAvailable reliable receive
      (parameter FederationTime))
    (class FlushQueueRequest reliable receive
      (parameter FederationTime)
```

```
               )
             ) ;; end Service
           )   ;; end Report
         )     ;; end Federate
       )       ;; end Manager
    )          ;; end InteractionRoot
  )            ;; end interactions
(ContextProviders
   (CP 11 MotionSensor
       (attribute X reliable timestamp)
       (attribute Y reliable timestamp)
       (attribute Z reliable timestamp)
       (attribute ID reliable timestamp)
   )
   (CP 22 TemperatureSensor
       (attribute Temperature reliable timestamp)
    )
)              ;; end FED
```

Creation of FED file manually is a complex process. To facilitate the creation of FED file and specification of HLA FOM and SOM, DMSO has provided the tool named “Object Model Development Tool” (OMDT). OMDT automate the process of creation of FED file.

OMT describes the data that is exchanged. Since the data that is exchanged is machine dependent therefore data is converted into the format appropriate for network. This process is known as parameter marshaling. Many systems define the machine independent representation of the data such as external data representation (XDR) to resolve data representation and transmission issues.

Our message format is simple. Each target that is detected by the RADAR is given a unique ID in the range from 0-999. Each of the fighter jet transmits its latitude, longitude and height in a 12 byte message described as follows:-

| 4 bytes | 4 bytes | 4 bytes |
|---|---|---|
| latitude | longitude | Height |

**Table 1: Fighter Jet Data Format**

When the target is detected by the RADAR, it is assigned a unique ID and upon confirmation it is passed to the operator of man-in-the-loop gun. The message format is as follows:-

| 2 bytes | 4 bytes | 4 bytes | 4 bytes |
|---|---|---|---|
| Target-ID | latitude | longitude | Height |

**Table 2: Detected Target Data Format**

## 3.3. Communication Mechanism

There are number of options available for data transmission/reception between federates distributed across multiple machine. These are:-

1. Distributed Component Object Model (DCOM)
2. Microsoft Messaging Queue (MSMQ)
3. Remote Procedure Call (RPC)
4. Sockets

### 3.3.1. Distributed Component Object Model

RTI, the central component of HLA based simulation model, can use DCOM for communication. DCOM is intended to provide distributed object services i.e. client and server may reside on different computers with client requesting the services from server remotely. The DCOM mechanism redirects all request to a 'server' that creates an instance of the object and passes the reference back to the client. The client can then invoke the methods on this object. This method cannot be used in our implementation because we do not want new instance of RTI to be created at each request. Although there exist mechanisms that allow for different applications to join to a single component but there are other issues like threading and marshalling, which make it more difficult to smoothly implement RTI in this fashion.

The RTI components can also be implemented as separate distributed services. But it has a drawback that more network bandwidth is required to transfer information between different RTI components then is required to execute those services.

### 3.3.2. Microsoft Messaging Queue

The second option that can be used by the RTI for communication between federates is Microsoft Messaging Queue. This is very enticing. This enables to establish a single queue where messages can be stored and retrieved. The idea was to send the service requests as messages and then let the RTI manage the queue and retrieve messages from it and then execute the services. But this has a lot of overheads and it results in performance degradation.

### 3.3.3. Sockets

The third option is to use low level TCP/IP sockets. TCP/IP is a specification of computer networks protocol which defines a set of rules to enable computers to communicate over the network. The rules contain information about message formatting, addressing and routing mechanism.

TCP/IP is implemented as a set of four layers which are as follows:-

1. Application Layer
2. Transport Layer
3. Internet Layer
4. Link Layer

Application uses the functionality of the layers by creating sockets.

### 3.3.4. Remote Procedure Call

The last option is to use RPC (Remote Procedure Call) for communication.

There are two mechanism used for RPC:-

1. Doors
2. Sun RPC

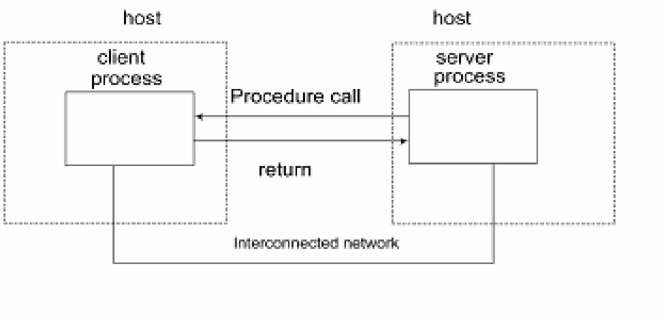


**Figure 9: Remote Procedure Call**

Doors provide the mechanism for RPC. Doors are identified by a descriptor within a process (client or server) and pathnames outside the process. A server creates a door by calling ‘door_create’, whose argument is a pointer to the procedure that will be associated with this door, and whose return value is a descriptor for the newly created door. The server then associates a pathname with the door descriptor by calling ‘fattach’**.** A client opens a door by calling ‘open’**,** whose argument is the pathname that the server associated with the door, and whose return value is the client descriptor for this door. The client then calls the procedure by calling ‘door_call’**.**

Sun RPC is another mechanism for RPC. When we require network communications among various pieces of the application, most applications are written using explicit network programming, that is, direct calls to either the socket API or the XTI API. However an alternative way for writing a distributed application using implicit network programming does exist. The calling procedure (the client) and the process containing the procedure being called (the server) can be executing on different hosts. The fact that the client and server are running on different hosts, and that network I/O is involved in the procedure call, is far the most part transparent. Sun RPC uses XDR, the External Data Representation standard, to describe and encode the data. XDR is both a language for describing the data and a set of rules for encoding the data.

Following is the steps executed in Remote Procedure Call:-

1. The server is started and it registers itself with the port mapper software on the server. The client is then started, which contacts the port mapper on the server host to find the server’s ephemeral port. Client then establishes TCP connection with the server.
2. The client calls a local procedure, called the client stub. The stub packages the arguments to the remote procedure into some standard format, and then builds one or more network messages. The packaging of client’s argument into a network message is termed as marshalling.
3. These network messages are sent to the remote system by the client stub by using TCP/IP protocol.

4. A server stub procedure on receipt of request from client, un-marshals the arguments from the network messages.
5. The server stub invokes a local procedure passing it the argument that it received from the client. When the server procedure is finished, it returns to the server stub.
6. The server stub marshals the return value and then sends back the message to the client.
7. The client stub reads the network messages from the local kernel.
8. After possibly converting the return values, the client stub finally returns to the client function.

### 3.3.5. Comparison of Communication Mechanism

The comparison of communication mechanism is given in the table below:-

| | Sockets | DCOM | MSMQ | RPC |
|---|---|---|---|---|
| Speed of execution | Fast | Slow | Slowest | Slow |
| Programming Effort | Highest | Medium | Medium | Medium |
| Learning required | Less | More | More | Medium |
| Network Bandwidth require | Less | More | More | More |

**Table 3: Communication Mechanism Comparison**

In our implementation we use sockets because of the following reasons

a. It gives us more control of our design
b. It does not make us dependent on a technology. Socket APIs are available for number of platforms
c. It allows us more flexibility in our design
d. It makes our RTI truly platform independent. As TCP/IP sockets can communicate on any platform.

## 3.4. RTI Services

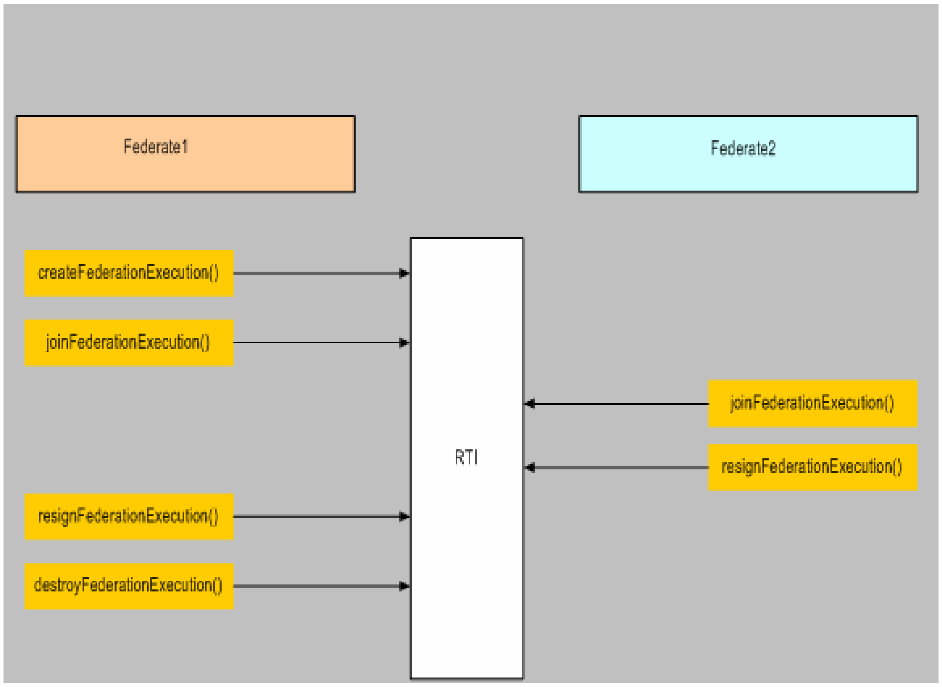
RTI provides services which can be grouped into six categories:-

1. Federation Management
2. Declaration Management
3. Object Management
4. Ownership Management

5. Data Distribution Management
6. Time Management

### 3.4.1. Federation Management

"Federation Management" refers to the creation, dynamic control, modification and deletion of federation execution [26].



Federate1
Federate2
createFederationExecution()
joinFederationExecution()
joinFederationExecution()
RTI
resignFederationExecution()
resignFederationExecution()
destroyFederationExecution()

**Figure 10: Federation Management Life Cycle**

There are three types of federates in our Air Defense Simulation. Gun Federate, RADAR federate and Fighter Jet Federate. Fighter Jet Federate is instantiated when the user specifies the flight trajectory of the fighter jet on the map. Similarly gun federate and RADAR federate is instantiated when the user specify the location of gun and RADAR on the map respectively. Operator can specify the flight trajectory, gun location and RADAR location in any order. These federate on start up join the federation. The first federate that starts also creates the 'Air Defense Federation'.

### 3.4.2. Declaration Management

Federate use 'Declaration Management' service provided by the RTI to declare their intention to generate and receive information. A federate must invoke appropriate declaration management services before it register object instances, update instance attribute values, and send interactions [26]. Federate sometimes also uses 'Data Distribution Management' service along with 'Declaration Management' service to declare their intention to receive information. Declaration management service of HLA includes publication, subscription and supporting control functions.

'Declaration Management' service can be best illustrated by the diagram below:-

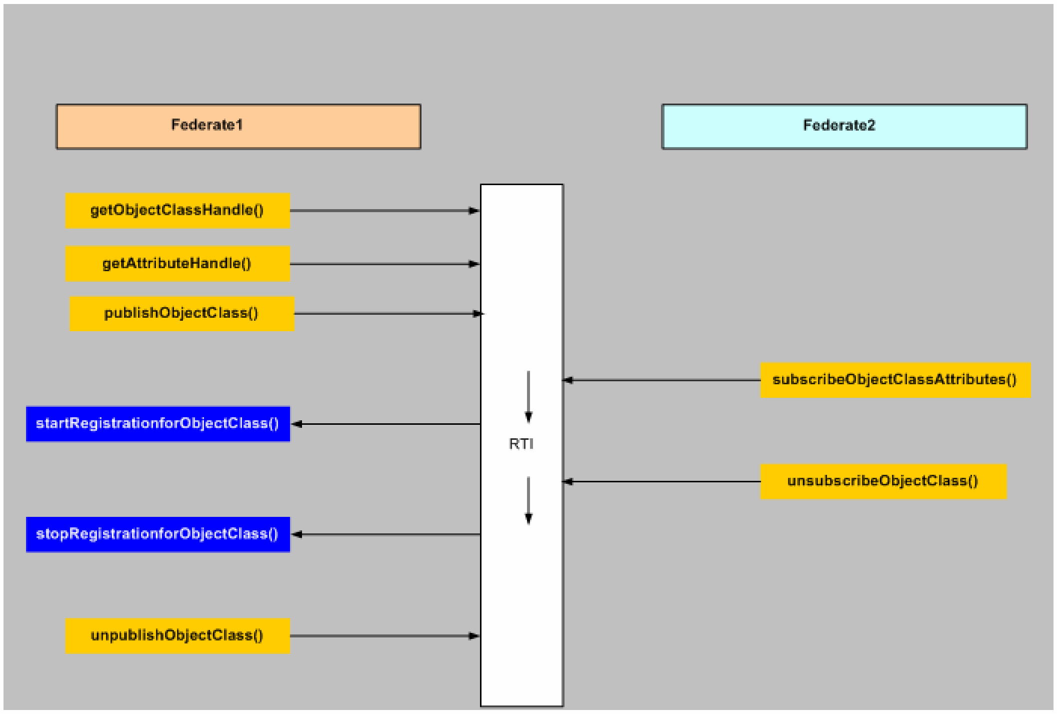


**Figure 11: RTI Declaration Management Service**

Each federate identifies its publication and subscription interests to the RTI LRC using the RTIAmbassador methods subscribeObjectClassAttributes() and publishObjectClass().The RTI signals a federate to start registration for object classes only when there is another federate who is interested in receiving the information. If no federate has shown interest in the information published by the federate, then that information is not put on the network.

Following table list the details of objects published or subscribed by federates in the Air Defense simulation.

| | RADAR | Gun | Fighter Jet |
|---|---|---|---|
| **Publish** | Detected target information | None | Latitude, Longitude and Elevation |
| **Subscribe** | None | Target Information | None |

**Table 4: Object Subscription and Publication Table**

Similarly the following table lists the interactions generated by the Air Defense Simulation:-

| | RADAR | Gun | Fighter Jet |
|---|---|---|---|
| **Publish** | Target Detected | Target Hit | Target Engaged |
| **Subscribe** | Target Hit | Target Detected | Target Engaged |

**Table 5: Interaction Publication and Subscription Table**

## 3.4.3. Object Management

'Object Management' includes instance registration and instance updates on the object production side and instance discovery and reflection on the object consumer side. Object management also includes methods associated with sending and receiving interactions, controlling instance updates based on consumer demand, and other miscellaneous support functions [8]. 'Object Management' service can best be illustrated by the diagram below:-

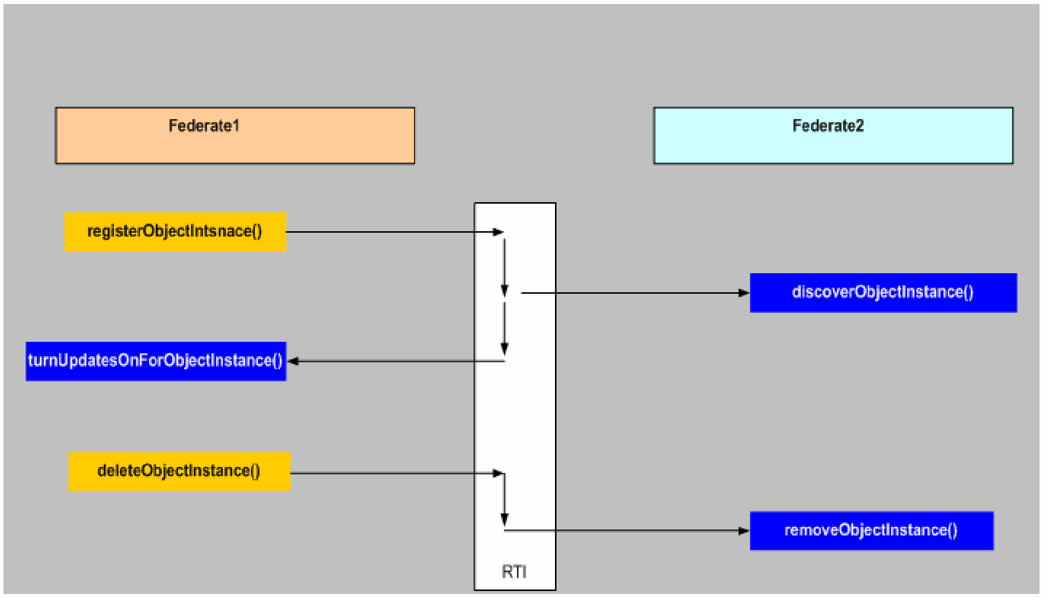


**Figure 12: RTI Object Management Service**

The RTIAmbassador method registerObjectInstance() inform LRC about the new object instance. Registration introduces an object instance to the federation. Updating of the values require other methods. To update the value of attribute the RTIAmbasssador method updateAttributeValues() is used. The federate who has previously discovered the new object receives the value by the FederateAmbassador callback method reflectAttributeValues ().

As stated previously in the section ‘HLA Object Management Template’, each federate is responsible for any data marshalling (encoding). The LRC does not enforce any encoding and does not know anything about contents. It only knows about name of object classes and their handles.

Object attribute updates and interactions are conveyed between federates using either ‘reliable’ or ‘best effort’ transportation scheme.

### 3.4.4. Ownership Management

'Ownership management' service of the RTI is used by federates and the RTI to transfer ownership of instance attributes among federates. The ability to transfer ownership of instance attributes among federates shall be required to support the cooperative modeling of a given object instance across a federation [26]. The ownership exchange among federates take place using either "Push" and/or "pull" model.  A federate can give away responsibility for one or more attributes of an object instance or can take ownership of an object instance. In case of Air Defense Simulation ownership of object is held by the object that initially created it.

### 3.4.5. Time Management

Accurate notation of time is important for distributed simulation. HLA based simulation can be categorized into two types on the basis of type of time management. Simulation can be "scaled real time simulation" or "as-fast-as-possible simulation". In "scaled real time simulation" where wall-clock time and simulation time have a linear relationship expressed as:-

$$T=S*W$$

where W is duration in wall-clock time, S is the scale factor and T is the duration in simulation time. When S=1 simulation is said to be a real-time simulation. Man-in-the loop and interactive simulation fall in this category.

In the simulation it is important to have a proper protocol for time management and synchronization.  Some protocol like TCP/IP provides inherent synchronization at the transport layer whereas others like UDP/IP do not provide synchronization.  Since HLA is independent and does not depend on other protocols for time management, it has its own mechanism for synchronization.

#### 3.4.5.1. Time Management in HLA

HLA federation has a concept of logical time which is independent of the local time of federates. There are four basic mechanisms for time management:-

1. Event Driven

2. Time Stepped
3. Parallel Discrete Event Simulation
4. Wall clock time driven

In ‘Wall clock time driven’ simulation mechanism, simulation time is derived from wall clock time. Wall-clock-time driven federates do not require that events be processed in time-stamp order. These simulations usually have hard and/or soft real-time constraints. Man-in-loop simulations use ‘wall-clock time driven’ management policy. In order to ensure a real-time behavior, they employ ‘clock synchronization’ and ‘time compensation’ algorithm.

Unlike message ordering and time-stamping requirements that are largely independent of the goals of the federation and what is being simulated. ‘Clock Synchronization’ and ‘time compensation’ are highly dependent on federation objectives and details of the model. Scheduling algorithms require detailed information concerning the computations performed within the federate, and thus are not well suited for implementation within the RTI. Time compensation techniques require information concerning the semantics of what is being simulated. Such information is not available within the RTI.

Clock synchronization is needed because hardware clocks in different (and geographically distributed) processors drift relative to one another. This can lead to serious problems in the distributed simulation if differences are large. The most used clock synchronization solution on the internet is the Network Time Protocol (NTP) which is a layered client-server architecture based on UDP message passing. NTP is a hierarchical protocol in which nodes attached to highly accurate time sources such as radio clocks, atomic clocks, and GPSs (Global Positioning Systems), called in NTP parlance stratum ones, share time among them and provide time to other NTP servers over the network.

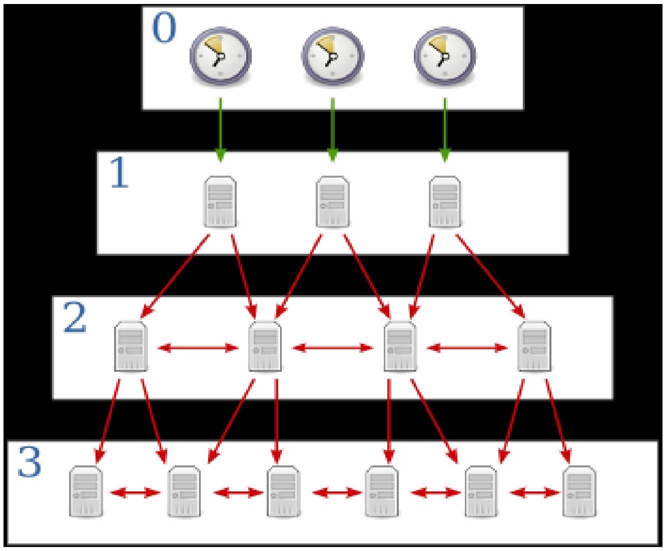


**Figure 13: Time Distribution**

It is important to have this kind of time synchronization in a man-in-loop simulation because usually such systems are coupled with a decision support system. These systems maintain logs for the replay of the scenario. In such cases if the system time is different, it become very difficult to correlate information needed in support of critical decision required for mission planning and execution.

Also for real time system behavior, time compensation is needed to account for the network delay. Network delay can be expressed as:-

$$\boxed{d_{\text{nodal}} = d_{\text{proc}} + d_{\text{queue}} + d_{\text{trans}} + d_{\text{prop}}}$$

Where

$d_{\text{proc}}$ = processing delay typically a few microsecs or less

$d_{\text{queue}}$ = queuing delay depends on congestion

$d_{\text{trans}}$ = transmission delay  L/R, significant for low-speed links

$d_{\text{prop}}$ = propagation delay from few microsecs to hundreds of msecs

Although queuing delay is an important delay factor in the packet switched network. In the simulation we assume that there is no queuing delay.

### 3.4.6. Data Distribution Managament

Data Distribution Management (DDM) services may be used by joined federates to reduce both the transmission and the reception of irrelevant data. Whereas Declaration Management (DM) services provide information on data relevance at the class attribute level, DDM services add the capability to further refine the data requirements at the instance attribute level [15].

## 3.5. Scenario Generator

Our scenario generator allows creation of various scenarios. It includes creation of various targets, platforms (both fixed and moving), command and communication post, sensors and decision support system.

### 3.5.1. Simulating Environment

For the current implementation, it is assumed that environment is static.

### 3.5.2. Simulating Fixed Wing Aircraft

Fixed wing aircrafts are governed by the laws of physics. The simplest model of aircraft moves forward, and maneuvers within the three rotational degree of freedom, i.e. pitch, roll and yaw. These reflect the main effect from changing the rudder, elevator and ailerons on a traditional aircraft and together with the thrust they constitute the primary flight controls or actuators.

Aircraft has a variety of interoceptive sensors and exteroceptive sensors. The interoceptive sensors are those sensors which measure the current state of the aircraft. Exteroceptive sensors are sensors which measure or monitor the environment outside the aircraft. Exteroceptive sensors include On-board RADAR, Forward looking infrared camera.

For the simplicity we assume that aircraft starts at the particular location. It is further assumed that aircraft change it location linearly according to the dead-reckoning algorithm.

### 3.5.3. Simulating Guns

The scenario generator allows creation of simulated guns node. The parameters of the guns can be specified while creating a node. Typical parameters of the gun are:-

a. Range of Gun
b. Mounting of Gun
c. Caliber of Gun
d. Rate of fire
e. Max Speed of Target that can be engaged
f. Sensors attached to the gun
g. Operating modes of the gun (automatic/manual)
h. Type of ammunition
i. Training speed of the gun
j. Elevation speed of the gun
k. Elevation range
l. Traverse of the gun
m. Blind Arc of the gun
n. Muzzle velocity of the gun
o. Misfire probability
p. Fire delay
q. Stabilized platform (yes/no)

### 3.5.4. Simulating RADAR

The scenario generator allows creation of RADAR node. There are various types of RADAR classified on the basis of their purpose and usage. Typical parameters of interest are specified below:-

a. Range Scale
b. Scan Rate
c. Sector or full scan
d. Operating Frequency
e. Pulse Repetition Frequency

f. Pulse Duration

g. Threshold for detection

h. Antenna rotation period

i. Set the Tilt of antenna

j. Gain

k. Use the IFF facility

l. STC (on /off)

m. Reduce Clutter

n. Reduce Rain effect

o. RADAR mode

    a. Long Range

    b. Short Range

    c. ASW

    d. Weather

    e. Beacon

p. Display Mode

    a. North Stabilized

    b. Relative bearing

## 3.6. Summary

The system consists of many components integrated together to form an ‘Air Defense Simulation’. The system ensures real time performance with time synchronization, data marshalling and filtering.

# CHAPTER – 4

## EXPERIMENT AND RESULT

## 4.1. Introduction

This chapter contains detailed description of the experiments conducted along with their results. The simulation is compared with that of the existing simulation and results were found to be comparable. Also log was maintained for each scenario so that it can be recreated.

## 4.2. Different Scenarios

Our system consists of RTI Network Centric Framework which displays information about all the federate in a federation. The information relates to federation management, declaration management, object management, time management and context providers.

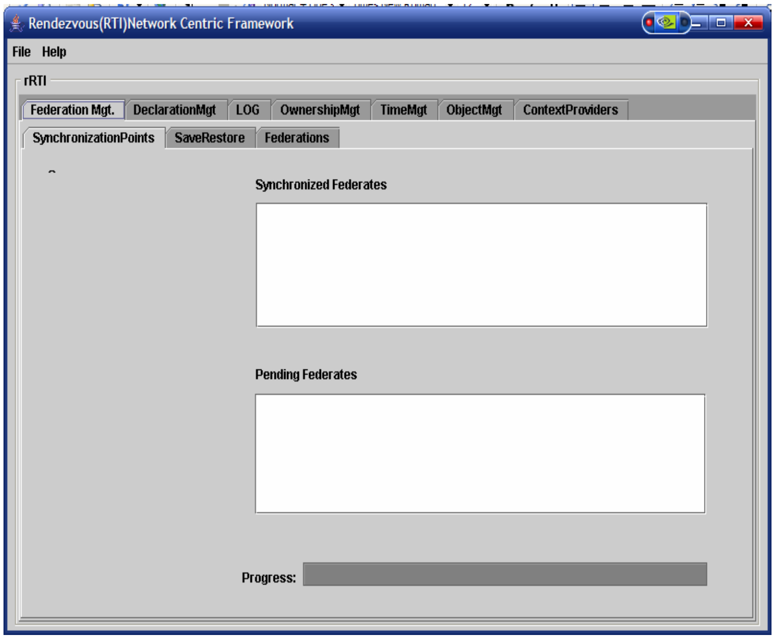


**Figure 14: RTI Network Centric Framework**

### 4.2.1. Single Target and Single Gun Scenario

In this scenario, we assume that aircraft start at the location. It changes it location linearly. It publishes the information about its current location (latitude, longitude and elevation) which is subscribed to by the RADAR. When the target enters within the range detectable by RADAR, it is detected and system generate track on it. The information is then passed on to the Gun. In this particular scenario we assume that there is a single target and the RADAR beam pattern is circular.

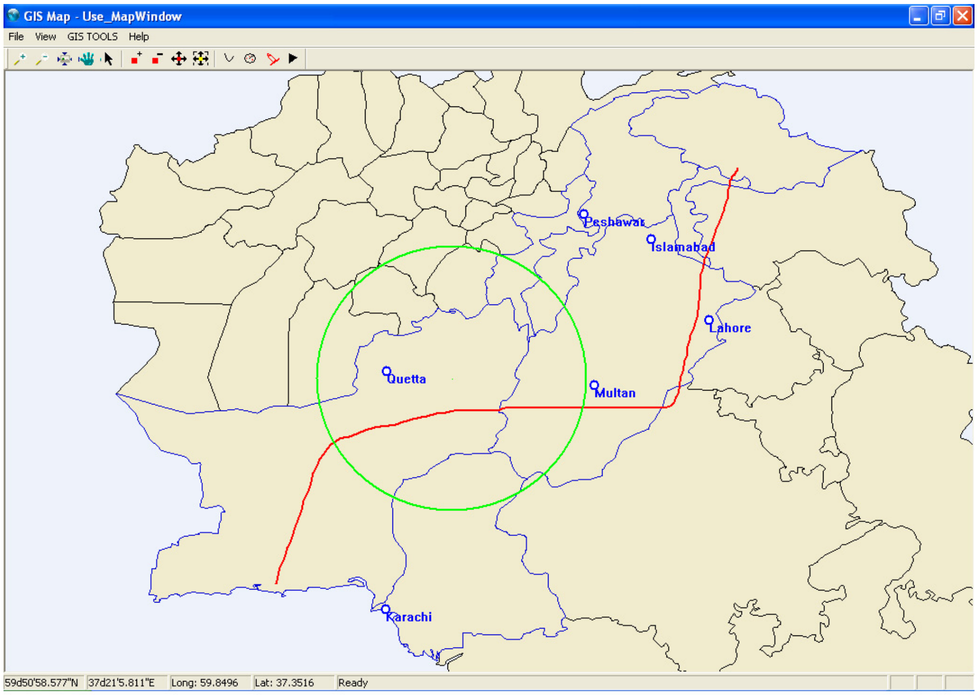
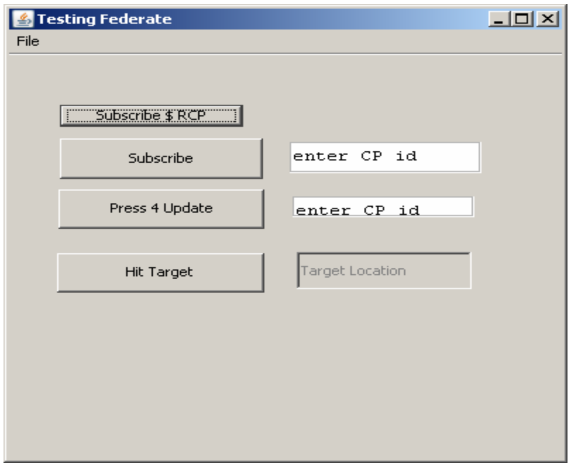


**Figure 15: Single Target Scenario**

Map display allows the operator to feed in the target trajectories. It allows for the placement of defense forces (RADAR) which is displayed superimposed on the map. The RADAR is a simple sector scanning RADAR. Furthermore, the display allows for the starting of simulation.

Once the target is detected by the RADAR, the information about the target is passed on to the gun federate.



enter CP id

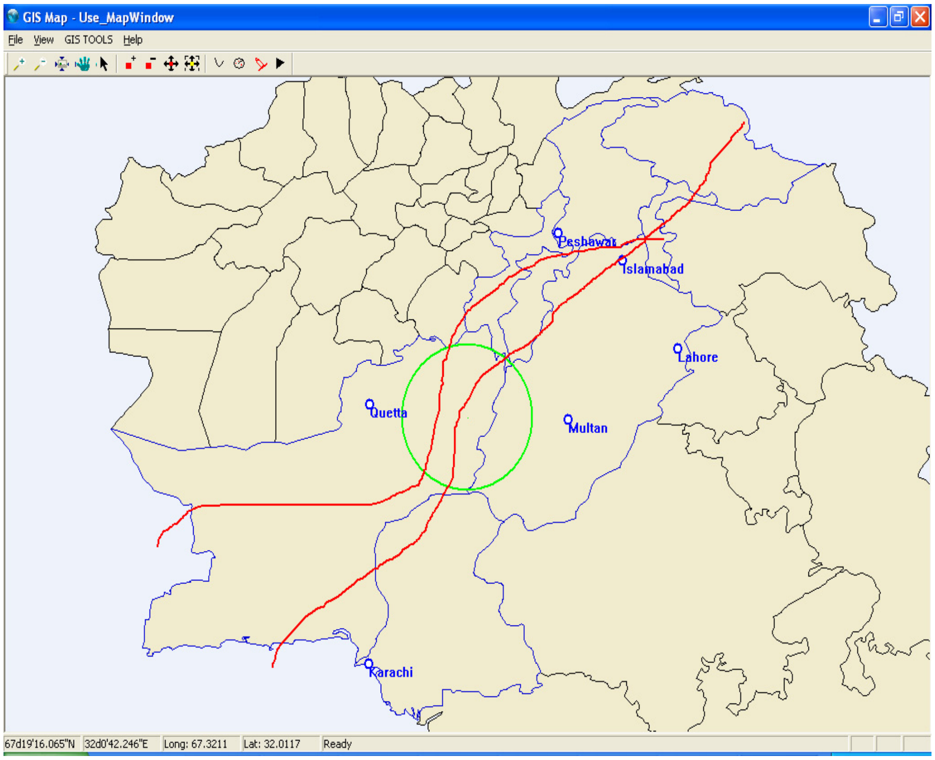
enter CP id

Target Location

Figure 16: Gun Federate

### 4.2.2. Multiple Targets and Single Gun Scenario

In this scenario there are two targets which are moving closing to each other. The targets are detected by single RADAR. Furthermore it is assume that there is a single gun within the area covered by RADAR.

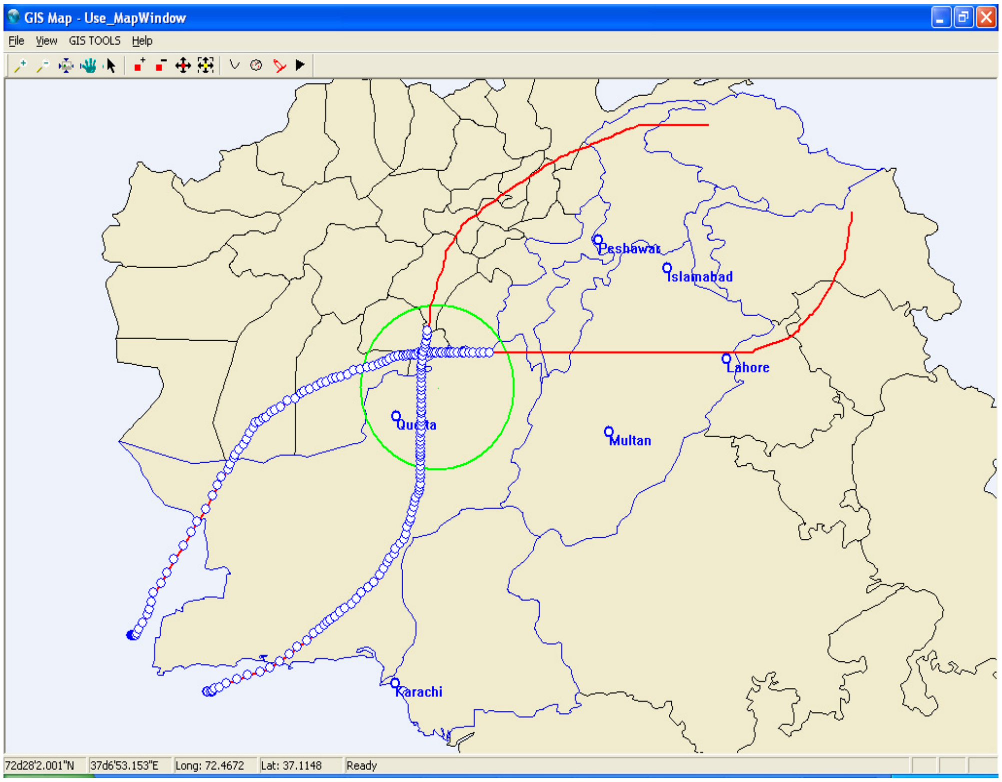


67d19'16.065"N 32d0'42.246"E Long: 67.3211 Lat: 32.0117 Ready

Figure 17: Multiple Target Scenario

### 4.2.3. Multiple Crossing Targets and Single Gun Scenario

In this scenario there are two targets which cross each other. They are detected by the RADAR. The target coordinates are given to the gun for engagement. In the multiple target scenarios it is observed that both target are detected by RADAR. However the first target that is engaged is the one that is closer to the gun.



72d28'2.001"N 37d6'53.153"E Long: 72.4672 Lat: 37.1148 Ready

Figure 18: Multiple Crossing Targets

### 4.2.4. Multiple Crossing Targets and Multiple Guns Scenario

The scenario is the same as the above scenario except that there are now multiple guns (two). Now the gun which is closer to target detects the targets. If the gun is placed close together, then was observed that 70% of the time, gun locks on to the same target and other target is missed by the gun.

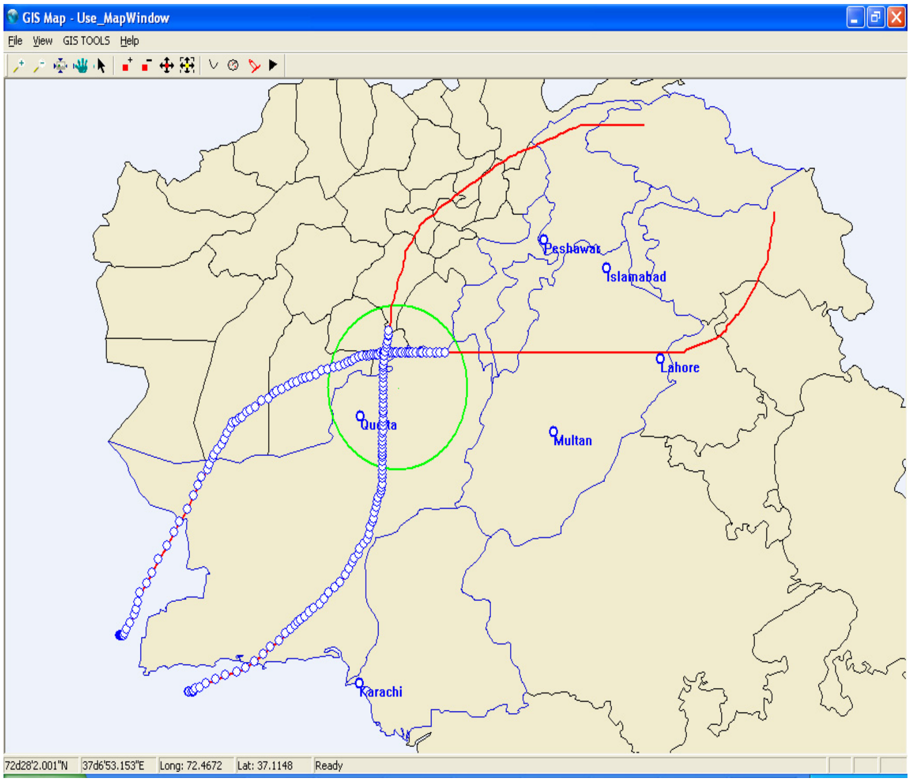


**Figure 19: Multiple Crossing Targets and Multiple Guns**

## 4.3. Summary

In this chapter the detailed results of the experiments were presented. The experiments were conducted using “RTI Network Centric Framework”, “Scenario Generator” and “Gun Simulator”. The experiment was repeated to verify the results.

# CHAPTER-5

# CONCLUSION

## 5.1. Introduction

This chapter is aimed to provide the key conclusions and future directions for the HLA based framework for man-in-the-loop simulation of Air Defense System.

## 5.2. Findings

Following are the main findings of this research.

### 5.2.1. Java and C++ Integration

"RTI Network Centric Framework" and all the libraries are developed in Java whereas the main simulation code uses C++. It was observed that JNI and java makes the system a little bit slower. The performance could be improved by using C++ entirely.

### 5.2.2. Time Management

In a real time man-in-loop simulation it is difficult to impose any time management since scenario can be modified at run-time. However in our case the scenario is pre-defined before simulation starts therefore time management policy can be defined. "RADAR Federate" act as a time regulating federate for the "Gun Federate", which is constrained.

## 5.3. Future Directions

### 5.3.1. Tracking Algorithm

The purpose of the research was to develop a framework so no real emphasis were laid on the tracking algorithm. However, tracking algorithms can be incorporated in the simulation.

### 5.3.2. Environment Modeling

Environment model can be incorporated in the system. Environment can greatly affect the RADAR performance.

### 5.3.3. Distributed Data Management

Since the simulation is limited in scope, the system does not implement the DDM service of HLA. DDM act as a filter on the updates. In our scenario the target continuously send its coordinate to all the RADAR federates known to the system. However restriction can be imposed like send the updates to the southern region when RADAR enters from the south side.

### 5.3.4. Decision Support System

Man-in-loop simulation usually has an integrated ‘Decision support system’ which evaluates the operator performance. In our particular scenario we only evaluate operator performance by the number of target detected by the RADAR and number of targets actually hit by the operator. More complex ‘Decision support system’ can be incorporated which allow us to replay the simulation.

## 5.4. Summary

We developed the man-in-loop simulation for Air Defense system using HLA. In our research we are able to identify key components for the developed of such system. Also key issues during the development of framework are discussed.

# REFERENCES

**[1].**“Missile system incorporating a targeting aid for man-in-the-loop missile controller” http://www.freepatentsonline.com/5605307.html

**[2].**“UK Tactical Data System Reference Guide”http://www.tdsrg.co.uk/v1c1.htm

**[3].**DARNOS http://www.kesem.com/DARNOS.asp?Validate=True

**[4].**Gil Tidhar, Michael Ling, Orly Shibi-Marr and Mario Selvestrel, **“***Human-in-loop simulation support to experimentation and concept development*”

**[5].**Heinze, C., Goss, S., Josefsson, T., Bennett, K., Waugh, S., Lloyd, I., Murray, G. and Oldfield, J., (2002) *“Interchanging agents and humans in military simulation”*, in AI Magazine, 23(2):37—47.

**[6].**Richard Hall, *“Path Planning and Autonomous Navigation for use in Computer Generated Forces”*, Scientific Report June 2007.

**[7].**Ling Rothrock, “*Using Time Windows to Evaluate Operator Performance*”, Department of Biomedical, Industrial, and Human Factors Engineering Wright State University

**[8].***RTI 1.3-Next Generation Programmer’s Guide Version 3.2*, US DoD Defese Modeling and Simulation Office

**[9].**F. Zhang and B. Huang, “*HLA-Based Network Simulation for Interactive Communcation System*”, First Asia International Conference on Modelling & Simulation, 2007

**[10].** T. Lee, S. Yoo, and C. Jeong, "*HLA-Based Object-Oriented Modeling/Simulation for Military System*", AsiaSim 2004, LNAI 3398,pp. 122-130,2005

**[11].** E. Biegeleisen, M. Eason, C. Michelson, and R. Reddy. "*Network in the loop using HLA, distributed OPNET simulations, and 3D visualizations*". In Military Communications Conference, 2005. MILCOM 2005. IEEE, volume 3, pages 1667--1671, Oct. 2005

**[12].** Sharif H.Calfee, "*Autonomous Agent-Based Simulation of an AEGIS Cruiser Combat Information Center Performing Battle Group Air-Defense Commander Operations*", Mater Thesis, Naval Postgraduate School ,March 2003.

**[13].** Osga, Glenn, et al., "*Design and Evaluation of Warfighter Task Support Methods in a Multi-Modal Watch Station*", Space and Naval Warfare Systems Center (SPAWAR), San Diego, May 2002, p. iii.

**[14].** Liebhaber, Michael J. and Feher, Bela, "*Air Threat Assessment: Research Model, and Display Guidelines*", p. 1.

***[15].*** *IEEE Std 1516-2000, 1516.1-2000, 1516.2-2000, Standard for Modeling and Simulation High Level Architecture (HLA)*

**[16].** Azzedine Boukerche and Kaiyaan Lu,"*A Novel Approach to Real-Time RTI based distributed simulation system*" in Proceeding of IEEE 38th Annual Simulation Symposim, 2005

**[17].** Bruzman, Don, Zyada ,Michael , Wasten, Kent and Macedonia, M. "*Virtual reality transfer protocol Design rationale*", Proceeding of the Sixth IEEE International workshop on enabling technologies: Infrastructure for Callaborative enterprises, Distributed system aspect of sharing a virtual reality, June 1997 pp 179-186

**[18].** *Real-Time CORBA 2.0: Dynamic Scheduling Specification*, November 2003, http://www.omg.org/

**[19].** L Georgiadis, R Guerin, V Peris and KN Sivarajan, "*Efficient network QoS provisioning based on per node traffic shaping*", IEEE/ACM Transaction on Networking ,1996

**[20].** Hui Zhao and Nicolas D. Georganas, "*HLA real time Extension*", Proceeding of Fifth IEEE International Workshop on Distributed Simulation and real-time Applications, Aug 2001.

**[21].** Douglos C. Schmidt, David L. Levine and Chris Cleeland, "*Architectures and Developing High Performance, real-time ORB Endsystems Advances in Computers*", Academic Press, Ed.,

**[22].** Gil Tidhar, Michael Ling, Orly Shibi Marr and Mario Selvestrel, "*Human-in-Loop Simulation Support to Experimentation and Concept Development*".

**[23].** Batchman, Loren E. (Solana Beach, CA), Foster and Carl G. (Tucson, AZ) ,"*Missile system incorporating a targeting aid for man-in-the-loop missile controller*", United States Patent 5605307, http://www.freepatentsonline.com/5605307.htm

**[24].** *"Hardware in the loop/Man in the loop*", Case Study by QualNet

**[25].** "*LabVIEW FPGA in Hardware-in-the-Loop Simulation Applications*", ftp://ftp.ni.com/pub/devzone/pdf/tut_3567.pdf

***[26].*** *High Level Architecture Interface Specification Version 1.3, U.S. Department of Defense*

**[27].** L.Neal Cosby, "*SIMNET-An Insider's Perspective*", http://www.sisostds.org/webletter/siso/iss_39/art_202.htm

**[28].** *"IEEE Standard 1278.1-1995 (and revisions)*", Standards Committee on Interactive Simulation (SCIS) of the IEEE Computer Society, Approved September 21, 1995.

**[29].** U.S. Department of Defense, ''*DoD Modeling and Simulation (M&S) Management,*'' Department of Defense Directive, Number 5000.59, January 4, 1994.

# AN HLA BASED MAN IN THE LOOP SIMULATION FRAMEWORK FOR AN AIR DEFENSE SYSTEM

**Syed Rauf ul Hassan**

**2005-NUST-MS PhD.CSE 05**

**Master Thesis**

# Scheme of Presentation

- Definition of Framework
- Overview of HLA
- Overview of “Man in Loop” and “Hardware in Loop”
- Functional Architecture of simulation
- Implementation
- Conclusion

# WHAT IS FRAMEWORK?

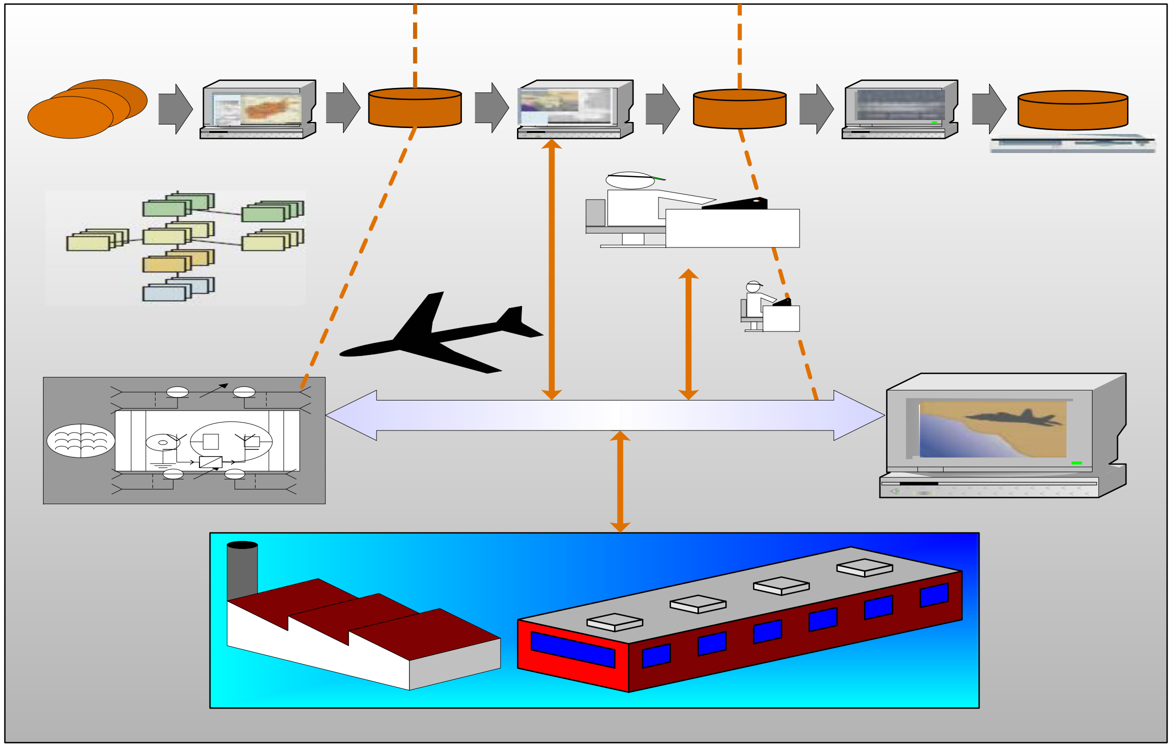
- A ***framework*** is a basic conceptual structure used to solve or address complex issues. This term is broadly used with the software.
- A **software framework**, is an abstraction in which common code providing generic functionality can be selectively overridden or specialized by user code providing specific functionality.
- Frameworks are similar to software libraries in that they are reusable abstractions of code wrapped in a well-defined API. Unlike libraries, however, the overall program's flow of control is not dictated by the caller, but by the framework.

# FRAMEWORK

- BFC is a RAD framework for developing database-centric distributed computing applications in a .NET environment.
- CNI (Compiled Native Interface) is a software framework for the GNU GCJ compiler which allows Java code to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in C++.
- Component-based Scalable Logical Architecture (CSLA) is a standard way to create robust object oriented programs using business objects, implemented using .NET.
- Java Native Interface (JNI) allows Java code running in the Java virtual machine (VM) to call and be called by native applications (programs specific to a hardware and operating system platform) and libraries written in other languages, such as C, C++ and assembly.
- Leonardi is an open source application framework for GUI applications
- Spring is an open source application framework for the Java platform.
- Symfony is a popular open source application framework for PHP Platform.
- CodeIgniter is a popular open source application framework for PHP Platform.
- Rails is a libre software application framework for the Ruby Platform.
- Zend Framework is a powerful and extensible application framework, with a loosely-coupled component library for PHP Platform.
- Twisted is an open source event-driven application framework written in Python for developing Internet applications

# Simulation

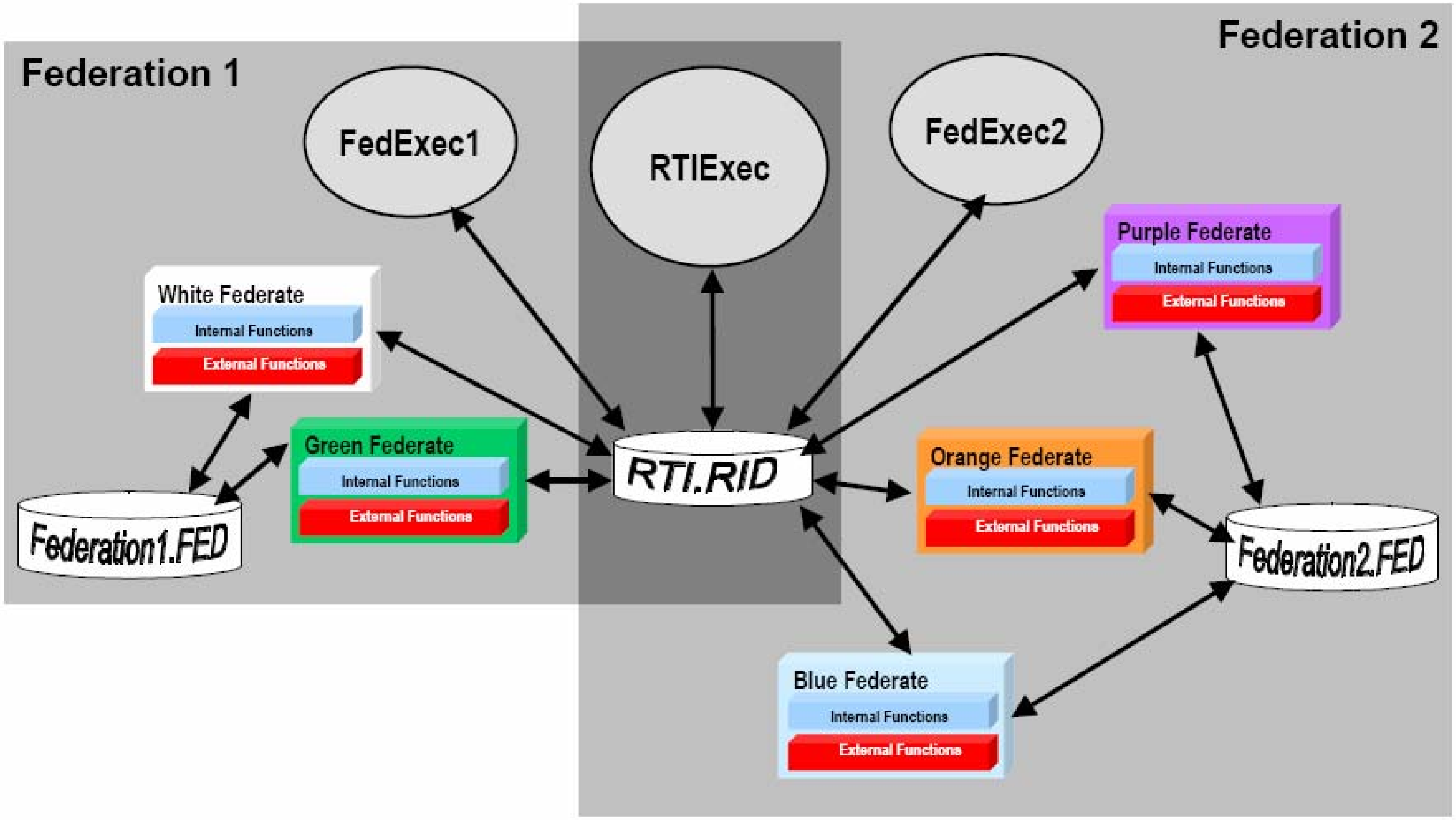
- **Simulation** is the imitation of some real thing, state of affairs, or process.
- Simulation is used in many contexts.
- Computer simulation is used for modeling natural processes



# INTRODUCTION HLA

- HLA is a general purpose architecture for simulation reuse and interoperability.
- Using HLA computer simulations can communicate with other computer simulations regardless of the platform. Communication between different simulations is managemed by Runtime Infrastructure (RTI )
- HLA is defined under IEEE Standard 1516.
- In our work we propose a simulation framework based on HLA.

# BIG PICTURE OF HLA



# WHY HLA?

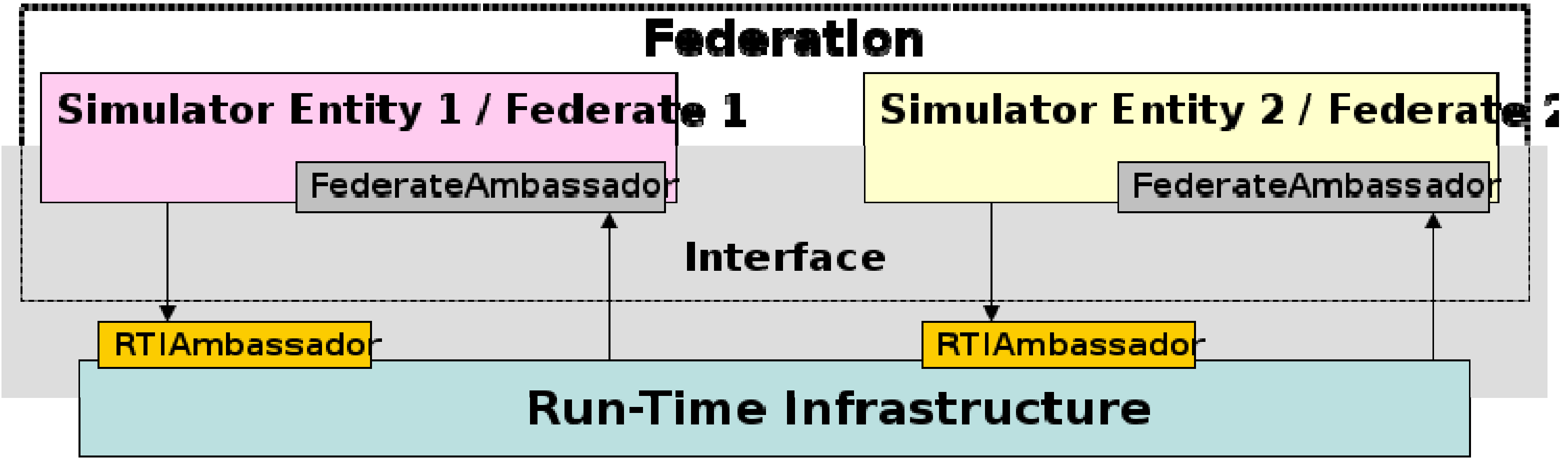
- There are many simulation frameworks available like
  - OMNET++ (discrete-event simulation development )
  - ModelSim (ASIC and FGPA simulation software )
  - QualNet (Wired and Wireless network simulation )
- HLA is developed specifically for the needs of defense industry.
- Initially developed by US DoD , it has become a standard for developing simulation. It has evolved to become IEEE 1516 standard.

# HLA OVERVIEW

- HLA consists of following components
  - HLA Rules
  - Interface Specification
  - Object Model Template

- HLA Rules
  - There are 10 rules out of which five are concerning the federates and five about federation
  - Ensure proper interaction of simulation in federation.
  - Describe the simulation and federation responsibilities

# HLA OVERVIEW



- Interface Specification
  - Defines the services provided by RTI (Run Time Infrastructure)
  - Identifies “callback” functions each federate must provide

# HLA INTERFACE SPECIFICATION

- RTI services separate simulation and communication
- HLA RTI services are classified into six types and can be used to manage individual simulation in a federation
  - Federation Management
  - Declaration Management
  - Object Management
  - Ownership Management
  - Data Distribution Management
  - Time Management

# HLA OVERVIEW

- Object Model Template
  - Provides a common method for recording information
  - Establish the format of key models
    - Federation Object Model
    - Simulation Object Model
    - Management Object Model

# Object Model Template (OMT)

- HLA OMT is used to specify FOM and SOM
- HLA Object Model shall consist of the following components:-
  - Object Model Identification Table
  - Object Class Structure Table
  - Interaction Class Structure Table
  - Attribute Table
  - Parameter Table
  - Routing Space Table
  - FOM/SOM lexicon

# Object Model Template (OMT)

- **<u>Object Model Identification table</u>** provide information that enable inferences to be drawn regarding the reuse potential of individual federates.

| Category | Information |
|---|---|
| **Name** | Rauf |
| **Version** | 1.0 |
| **Date** | 03-12-2008 |
| **Purpose** | Man in loop simulation framework Project |
| **Application Domain** | Air Defense Simulation |
| **Sponsor** | |
| **POC** | NUST College of E&ME |
| **POC Organization** | NUST |
| **POC Telephone** | 0519278050 |
| **POC Email** | rauf.hassan@gmail.com |

# Object Model Template (OMT)

- **Object Class Structure Table** define a set of relations among classes of objects from the simulation or federation domain.
- Each object class in object class structure table is followed by information about publication and subscription capabilities of object class.

| | **RADAR** | **Gun** | **Fighter Jet** |
|---|---|---|---|
| **Publish** | Detected target information | None | Latitude, Longitude and Elevation |
| **Subscribe** | None | Target Information | None |

# Object Model Template (OMT)

- Interactions are specified in “Interaction Class Structure Table” similar to the way object classes are specified in Object Class Structure Table.
- “Interaction class structure table” also includes capability of given type of interaction which may be initiates , senses, reacts and none.

|  | **RADAR** | **Gun** | **Fighter Jet** |
|---|---|---|---|
| **Publish** | Target Detected | Target Hit | Target Engaged |
| **Subscribe** | Target Hit | Target Detected | Target Engaged |

# Object Model Template (OMT)

- **Attribute Table** specifies the characteristics for attributes in the attribute table.
- It contains the following information
  - Object Class
  - Attribute Name
  - Data Type
  - Cardinality
  - Units
  - Resolution
  - Accuracy
  - Accuracy Condition
  - Update Type
  - Update rate/condition
  - Transferable/acceptable
  - Updateable/ reflectable
  - Routing Space

# Object Model Template (OMT)

- Parameters associated with interaction specified in “Interaction Class Structure Table” are specified in “Parameter Table”.
- Following characteristic are specified for each parameter
  - Interaction Class
  - Parameter name
  - Data Type
  - Cardinality
  - Units
  - Resolution
  - Accuracy
  - Accuracy Condition

# Object Model Template

- “Routing Space Table” is useful in “Distributed Data Management”. Routing space table is a multi dimensional coordinate system through which federate either expresses an interest in receiving data or declare their intention to send data.
- FOM/SOM Lexicon provides a mean for federations to document the definition of all terms utilized during the construction of FOMs and SOMs.

# INTRODUCTION MAN-IN-LOOP

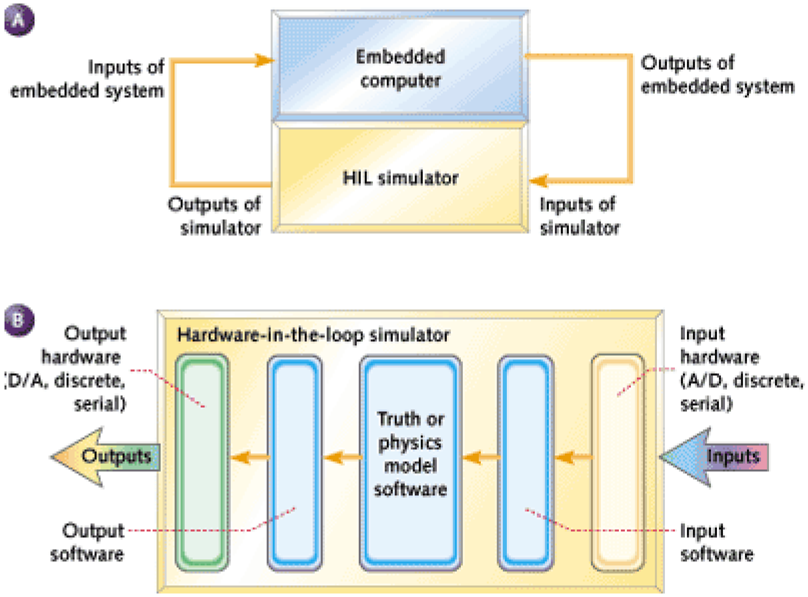
- In Man-in-loop or Interactive simulation one or more simulation module is controlled by human beings. There are also called as “live simulation”

# WHY MAN IN LOOP?

- To study human factors in the development of system.
- To train the operator

# HARDWARE IN LOOP

Although our man emphasis is man-in-loop, we discuss hardware in loop for completeness. A HILS is a device that fools your embedded system into thinking that it's operating with real-world inputs and outputs, in real-time. In the autopilot example, it fools the aircraft into thinking it's flying. (http://www.embedded.com/15201692)


A
Inputs of embedded system
Embedded computer
Outputs of embedded system
HIL simulator
Outputs of simulator
Inputs of simulator
B
Output hardware (D/A, discrete, serial)
Hardware-in-the-loop simulator
Input hardware (A/D, discrete, serial)
Outputs
Truth or physics model software
Inputs
Output software
Input software

# HARDWARE IN LOOP

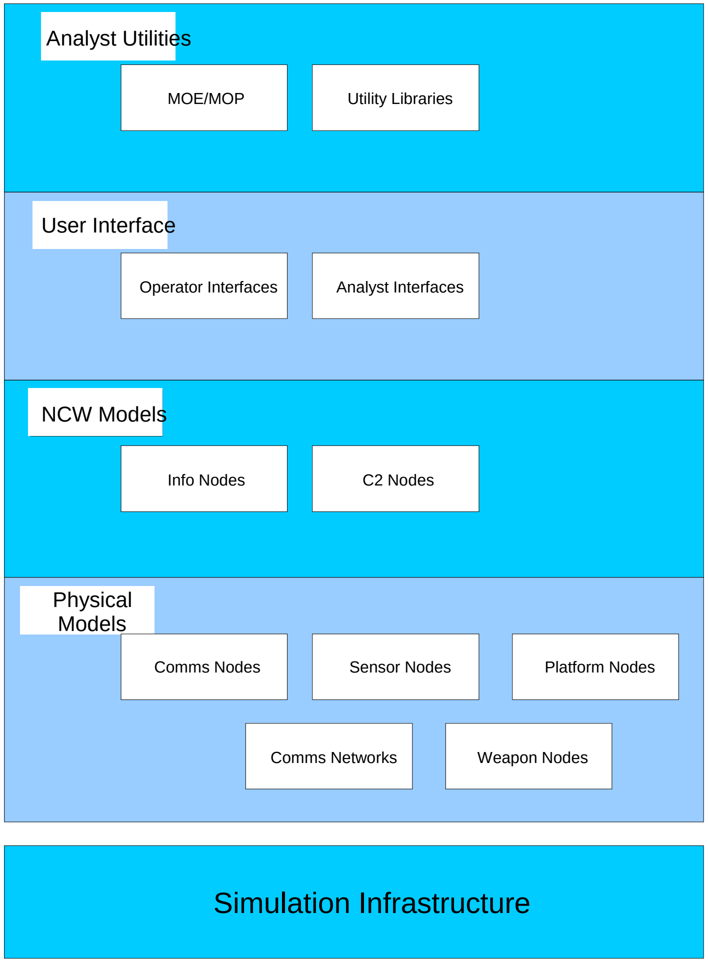
- Real Time HIL Simulation Framework is available at http://sourceforge.net/. It utilizes SysV IPC and Glade/GTK user interface .

# System Description

- The system consists of various air defenses like RADAR and guns
- RADAR detects the target and
- Guns engage the target which is detected by the RADAR
- Target may be a missile, fighter jet.

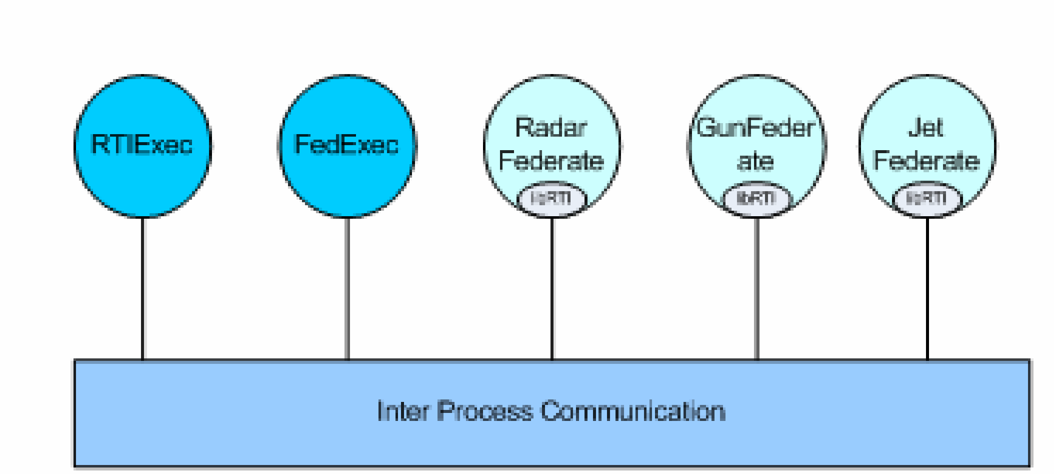
# Functional Architecture of man-in-loop Simulation

- The physical components model the physical characteristics of the NCW Players that exist within an NCW scenario.
- NCW component models the command and control and information sharing capabilities
- The user interface component is used to support HIL experimental activity.
- The Analyst Utilities component contains libraries/components to measure performance and effectiveness.



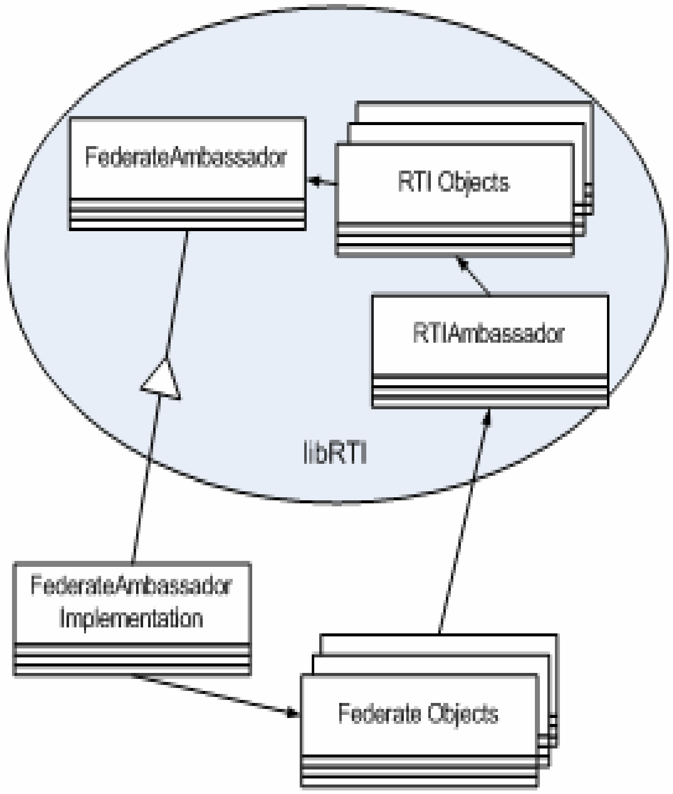
# Functional Architecture of man-in-loop Simulation

- RTIExec is a global process.
- FedExec manages federate. It is created by first federate joining the federation.
- Federate communicate by using IPC mechanism. It is allobrated further in next slides.



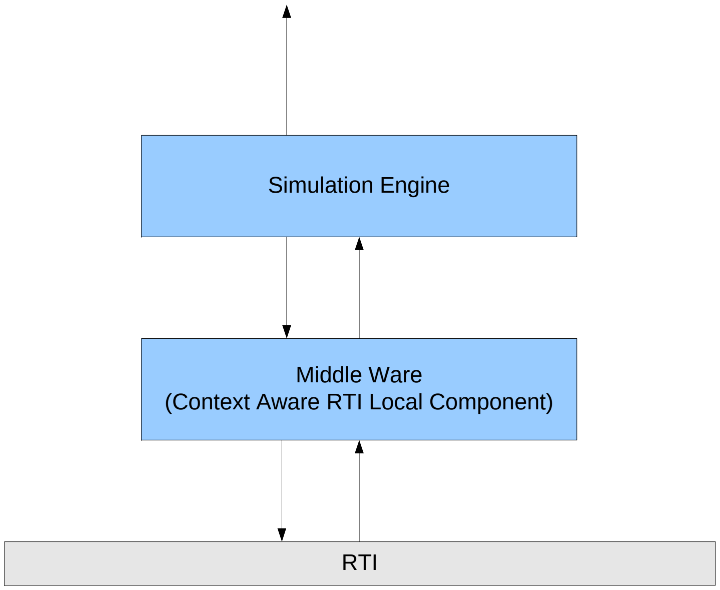
# Functional Architecture of man-in-loop Simulation

- Federate is depicted in the figure
- To create a new federate a class is derived from **Federate Ambassador**.
- The newly created class get the handle to **RTIAmbassador** and use it to invoke various services provided by RTI



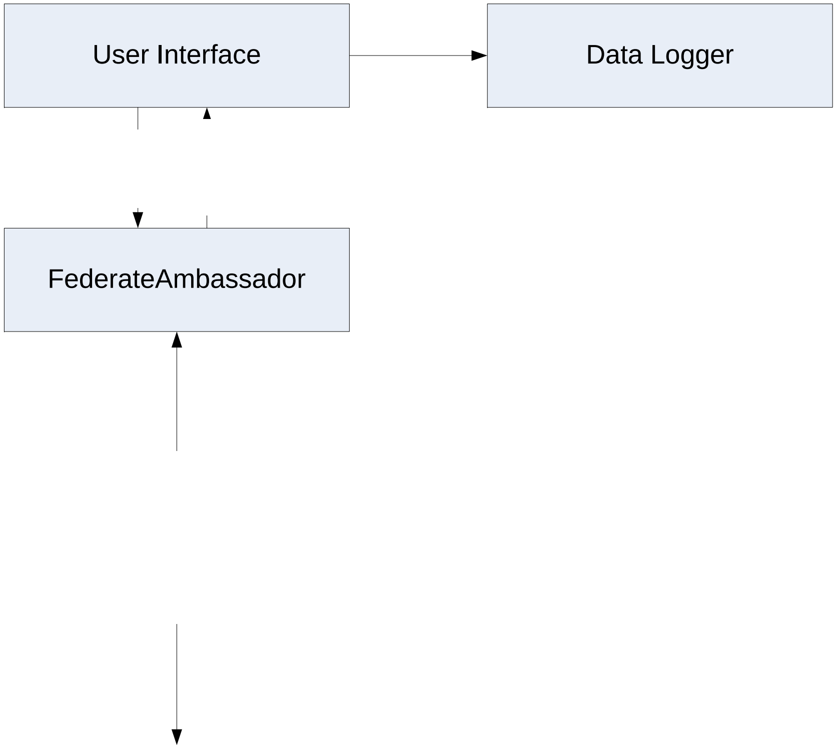
# Functional Architecture of man-in-loop Simulation

- Each federate consist of two components
  - Middle Ware undertaking all the communication with RTI
  - Simulation Engine



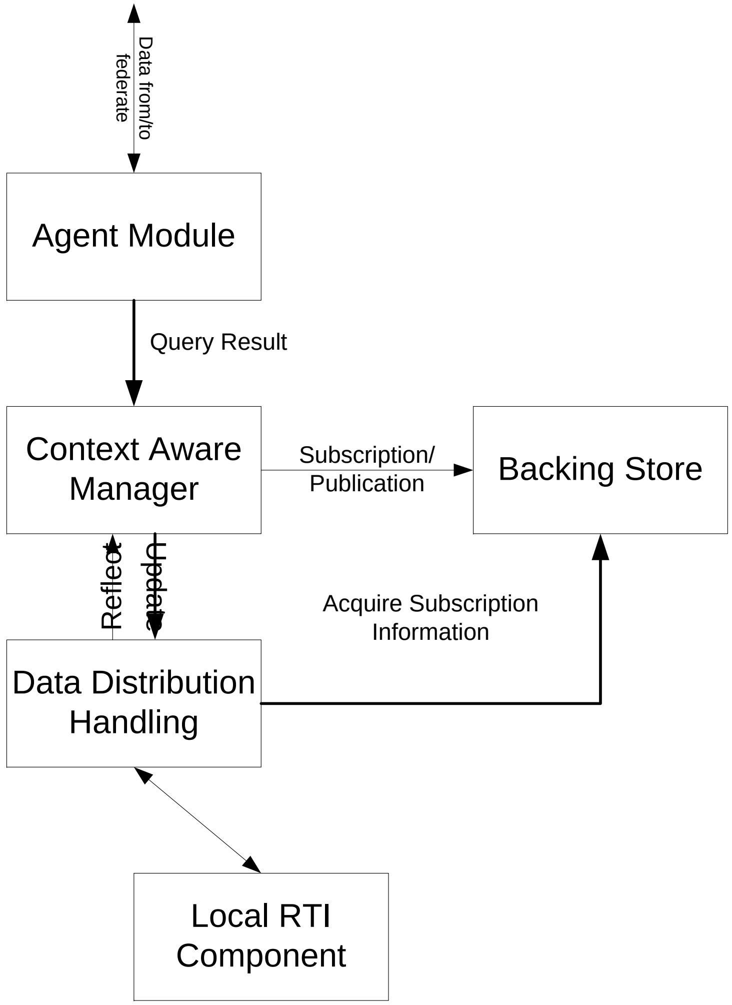
# Functional Architecture of man-in-loop Simulation

- Internal Architecture of Simulation Engine



# Functional Architecture of man-in-loop Simulation Middleware

- Internal Architecture of Middleware
- All the communication between Federate and RTI is done through Agent Module
- Context Aware manager receives subscription/publication request from Agent Module and send it to backing store. Update request are send to “Data Distribution Handling” Module.
- Data Received from RTI are send to the “Context Aware Manager” via “Data Distribution Handling” module by reflection.

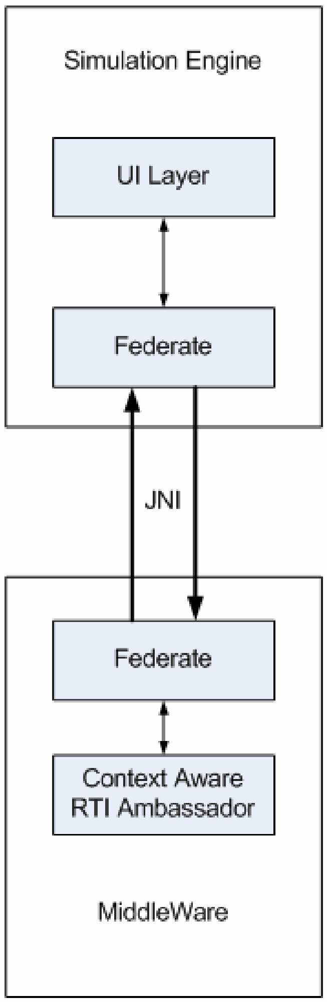


# Functional Architecture of man-in-loop Simulation

- The system consists of three types of interacting subsystems/federates
  - Fighter Jet
  - RADAR
  - Guns or firing system
- These modules are initiated on the user request by the "User Interface Module"
- "User Interface module" in "Simulation Engine" allows the following activities:-
  - Create the flight path for the Fighter Jet on the digital map
  - Specify the location of RADAR on the digital map
  - Specify the location of Air Defenses (Gun, Missiles) on the digital map
  - Start the Simulation

# Interfacing between Middleware and Simulation Engine

- Interface between middleware and simulation engine is depicted below for one Federate. Other Federate follows the same approach



# Communication Mechanism

- Each of the subsystem/federates can communicate by using:-
  - DCOM (Distributed Component Object Model)
  - MSMQ (MicroSoft Message Queue)
  - *Sockets*
  - RPC (Remote Procedure Call)

# Communication Mechanism-DCOM

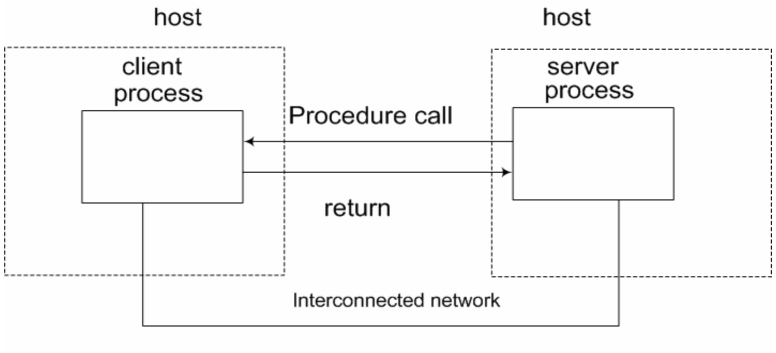
- DCOM is intended to provide distributed object services.
- The DCOM wire protocol transparently provides support for reliable, secure, and efficient communication between COM components such as Microsoft® ActiveX® controls, scripts, and Java applets residing on different machines in a LAN, a WAN, or on the Internet.
- DCOM mechanism redirect user request to a **server machine** which create instance of new object and passes the reference to client.
- We cannot create a new RTI instance when a request to join is made by Federate.

# Communication Mechanism-MSMQ

- The second option is to use MSMQ.
- MSMQ uses single queue where messages are stored.
- Service requests are send as message

# Communication Mechanism-RPC

- Remote Procedure Call is used to invoke the services on another machine in the network
- Two mechanism for Remote Procedure Call (RPC)
  - Doors
  - Sun RPC



**Remote procedure call**

# Communication Mechanism-Sockets

- Network Sockets are the most common method of transmission on network.
- TCP/IP is the most widely used protocol.

# Communication Mechanism

- The comparison is given in the following table

| | Sockets | DCOM | MSMQ | RPC |
|---|---|---|---|---|
| Speed of execution | Fast | Slow | Slowest | Slow |
| Programming Effort | Highest | Medium | Medium | Medium |
| Learning required | Less | More | More | Medium |
| Network Bandwidth require | Less | More | More | More |

- We use sockets because
  - It give us more control on design
  - Socket APIs are available on different platform.
  - It makes RTI platform independent

# DESIGN ISSUES

- Main issues that needs to be considered are
  - Time Management
  - Data Marshalling
  - Security

# OTHER DESIGN ISSUES:
# TIME MANAGEMENT

- Time Management Includes
  - Transportation Service
  - Time Advancement Service
- HLA support two kind of transportation services
  - Reliable
  - Best Effort
- Transportation service is specified in HLA Fed file for each class attribute and interaction. For example

```
 (class FighterJet
    (attribute latitude reliable timestamp)
     (attribute longitude reliable timestamp))
```

# OTHER DESIGN ISSUES: TIME MANAGEMENT

- Five message ordering mechanism are provided by HLA
  - Receive Order
  - Priority
  - Time Stamp Order
  - Causal Order
  - Causal and Totally Ordered

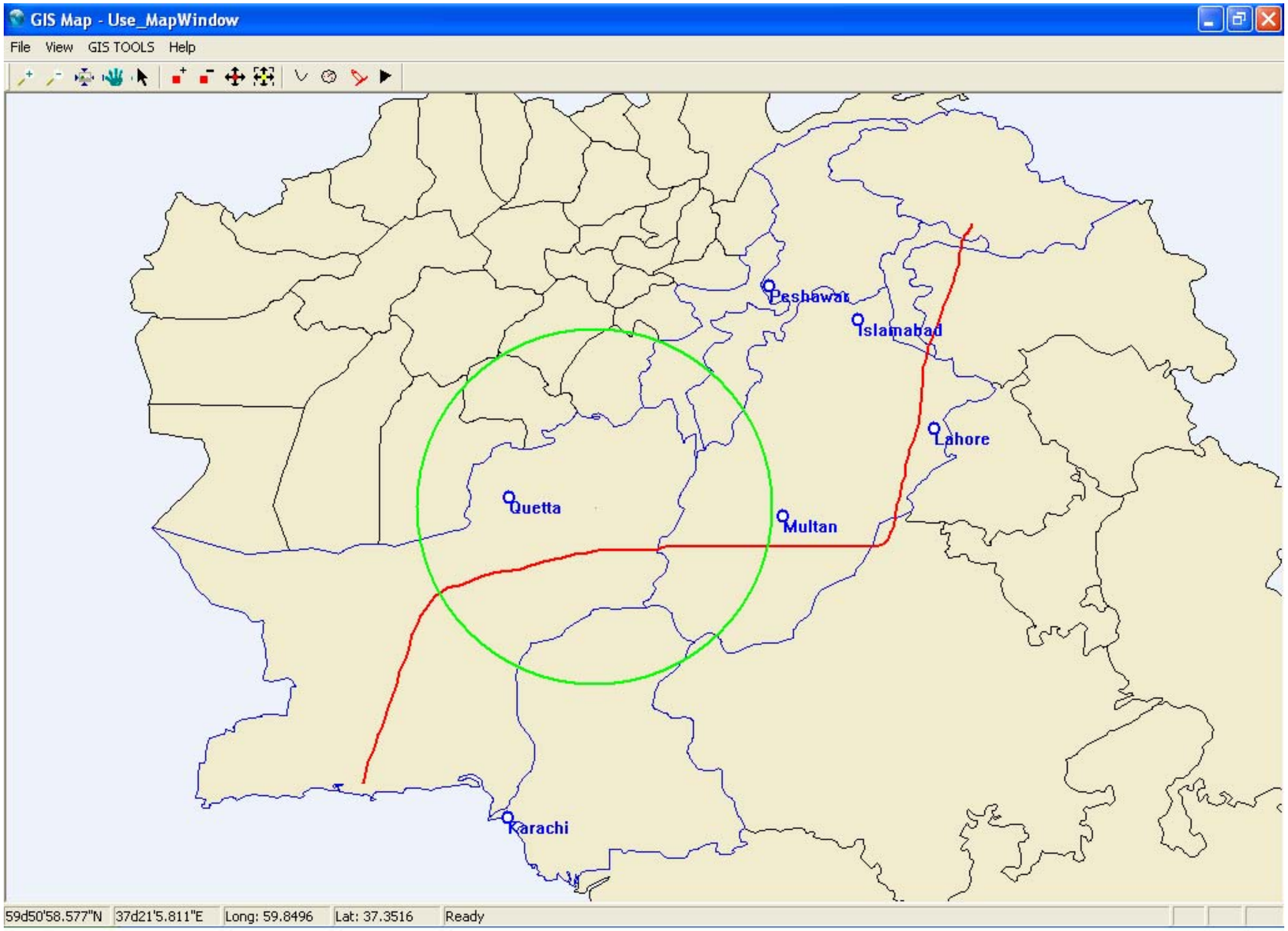
# OTHER DESIGN ISSUES: TIME MANAGEMENT

- The RADAR federate autonomously advances its own time without coordinating such advances with the RTI. “Human-in-the-loop” training federates and “hardware-in-the-loop” test and evaluation federates typically utilize this approach.

# OTHER DESIGN ISSUES: DATA MARSHALLING

- Data marshalling is not implemented by the RTI and HLA. So the federate has to implement it themselves.
- It involves converting the commands and data from sender into appropriate format to be received by the receiver.

# SCENARIO GENERATOR



# SCENARIO GENERATOR

- The system is evaluated for the number of guns and RADAR and fighter jets interacting with each other and with tactical decision support system through well defined interfaces.
- The most common format for data exchange can be
  - Link-1
  - Link 14
  - Link 11A
  - Link 11B
  - Link 16
- These formats defines the following:-
  - Message Format
  - Medium
  - Data Rate
  - Encryption

# SCENARIO GENERATOR

- The message format is specified and HLA based federates exchange messages in the same format as specified in the standard for such system
- Also the system is evaluated for a given data rate by using a logging facility of network monitoring tool (ethereal or pcap).

# IMPLEMENTATION

- The system presently have the following three types of federates
  - Scenario generator federate
  - RADAR federate
  - Gun or fire engine federate

- Scenario generates various targets like fighter jet, air to surface missile , surface to surface missile.
- Scenario generator may include the environment modeler. But at present it is not included
- For the purpose of demonstration scenario generates generate targets which have
  - Initial location (latitude, longitude)
  - Initial course
  - Initial Speed
  - Course
  - Speed

# IMPLEMENTATION

- RADAR federate implements a simple sector scan radar
- RADAR is operated by an operator which can create tracks on various targets. Targets are then handed over to the firing solution.
- In the implementation scenario that are discussed later three possibilities are discussed.
  - Place the actual gun in loop with either operator control or manual control
  - Simulate the gun

# IMPLEMENTATION SCENARIOS

- Scenario-1
  - There is federate which act as a Scenario generator. The main purpose of it is to generate various targets like jets and missile.
  - RADAR is simulated as an HLA federate in a simulation. Operator detects the target which is man-in-loop
  - Gun is simulated. Operator can see the virtual console where he can designate various targets.

# IMPLEMENTATION SCENARIOS

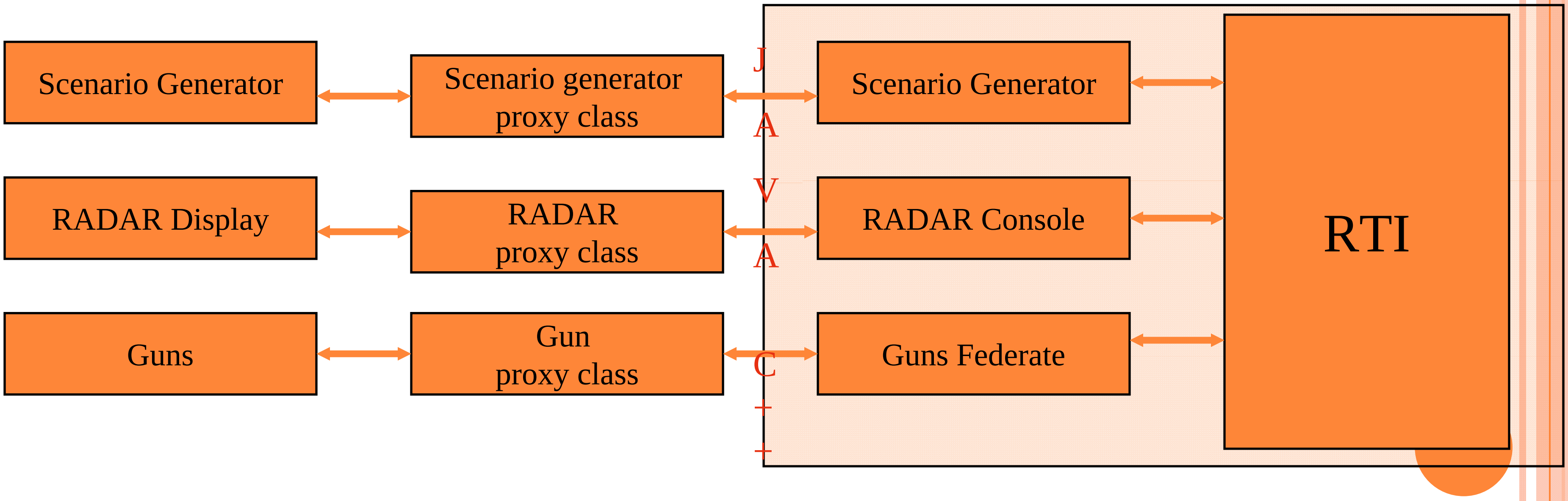
- Scenario-2
  - There is federate which act as a Scenario generator. The main purpose of it is to generate various targets like jets and missile.
  - RADAR is simulated as an HLA federate in a simulation which is man in loop
  - Gun is simulated as virtual federate interacting with actual hardware in loop gun. When target is detected on the RADAR screen, it is send to gun on the RS-232 interface

# IMPLEMENTATION SCENARIOS

- There are other scenarios like
  - Single Target Single Gun Scenario
  - Multiple Target and Single Gun Scenario
  - Multiple Crossing Target and Single Gun Scenario
  - Multiple Crossing Targets and Multiple Guns Scenario

# IMPLEMENTATION

- The system consist of RTI (Run Time Infrastructure) layer which is implemented entirely in Java. For RTI , Pitch portable RTI is used (http://www.pitch.se/products/pitch-prti/pitch-prti-overview.html)
- Federates are implemented in Java. They interact with the Graphics module implemented in C++ by way of JNI.

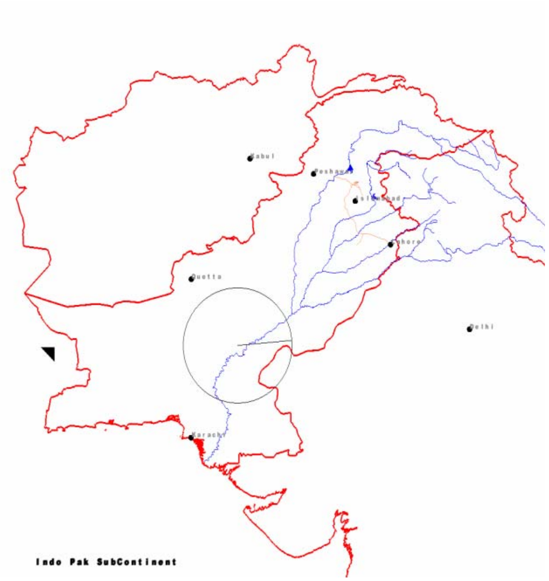


# IMPLEMENTATION

- Platform: Windows XP and compatible
- Java is used for an implementation of RTI
- OpenGL is mainly used for Drawing
- MapWindow (http://www.mapwindow.org/) is used for drawing the map of Sub Continent.
- Main routines are written in C++. They interact with Java through JNI

# IMPLEMENTATION

- Demonstration



RADAR Simulator

Gun Virtual Federate

Encarta Encyclopedia, Masterfile/Peter Christopher

Real Gun

# CONCLUSION

- There are various advantage of an HLA based man-in-loop simulation.
  - It allows for the operator training on various scenarios.
  - It also allows for testing of new weapons in laboratory. Hardware-in-loop allows for testing of interfaces. whereas man-in-loop allows for testing of the usability.

# FUTURE WORK

- The main emphasis of this research is to study the ‘HLA based man in loop simulation’. ‘Hardware in Loop’ scenario is not fully implemented. In future this scenario can be implemented.
- Target Tracking Algorithm
- Environment Modeler
- Distributed Data Management
- Decision Support System

# REFERENCES

- https://www.dmso.mil/public/transition/hla/
- http://www.embedded.com
- "Missile system incorporating a targeting aid for man-in-the-loop missile controller" http://www.freepatentsonline.com/5605307.html