International Journal of Database Theory and Application
Vol.7, No.5 (2014), pp.187-200
http://dx.doi.org/10.14257/ijdta.2014.7.5.14

An Object-Oriented Design Methodology Based on Object Service Tier (OST) Middleware in HLA Framework for the Distributed Simulation System Environment

Usman Sikander

usmansikander@yahoo.com

Abstract

Promising the developers with the facility of distributed collaborative development for complex simulation applications, HLA (High Level Architecture) provides a baseline supporting the reuse of capabilities available in different simulations with a significant reduction in cost and time. Along with improved execution process and reusability, the induction of object oriented model in the implementation design of HLA also enable to take the advantage of latest object oriented features making design, implementation and maintainability easier and at each level of federate development and execution.

The important areas to be addressed for design reconsideration consists of data exchange model and HLA communication layer. The data exchange model comprises of federates in federation and between runtime infrastructure and federation. The Federate Object Model (FOM) architecture is not completely object oriented, the induction of Object Service Tier (OST) middleware may offer a degree of FOM agility which is the ability of an application to adjust according to different FOMs (behaviors for Federates). Whereas, in HLA communication layer, customary HLA systems are based upon bidirectional call/callback interactions between federate. Several enhancements and changes anticipated in object oriented communication layer (OOP-COMM) introduced in Object Service Tier (OST) as compared to native procedures, such as communication mechanisms, data encoding, session handling, distributed environment and performance analysis.

The motivation behind the use of core object oriented modeling features and proposed Object Service Tier (OST) middleware is the reuse of legacy systems, features which may further enhance the integration of distributed simulation systems and extension types. So, this paper provides a multidimensional analysis of important design aspects of Object Service Tier (OST) middleware in HLA framework and devises some design constructs of Object Service Tier (OST) using object oriented model. This paper is intended to propose object-oriented model providing generalization through the Object Service Tier (OST) middleware in HLA framework for the distributed simulation system environment.

Keywords: *HLA, Middleware, Simulation Middleware Object Classes, Java, FOM, Code generation, OO-HLA, distributed computer simulations*

1. Introduction

1.1 Background and Introduction of HLA Architecture

HLA architecture is extended across a wide range of simulation areas, including education, training, examination and engineering. Definition of HLA includes major functional elements, interfaces, design rules, and provides a common framework within which specific system architectures can be defined. This definition is in accordance with the IEEE definition of

ISSN: 2005-4270 IJDTA

architecture for distributed computer simulations. The HLA does not specify an implementation, and also does not suggest the use of any particular software or programming language [2].

1.2 HLA Framework Artifact

Important vocabularies defining the basics for HLA Framework [2] are:

Federation: It is set of federates with a common *Federation Object Model*. The federation consist of a number of sub-systems that together with a *Runtime Infrastructure (RTI)* forms simulation model, *i.e.*, a simulator.

Federate: A federate can be a sub-system of a simulator or a whole simulator in a multi-simulator *Federation*.

Runtime Infrastructure (RTI): This is central kernel in the architecture of a simulator. All *Federates* communicate through the runtime infrastructure.

Object Model Template (OMT): A documentation standard for the consistency of data and interactions, used for communication between HLA *Federates*.

Federation Object Model (FOM): The common description of interactions to be communicated between *Federates* in a *Federation*. The *Runtime Infrastructure (RTI)* uses the FOM for routing the information between *Federates*.

Simulation Object Model (SOM): Each *Federate* provides *Objects* and *Interactions* to a *Federation*. The description of a *Federate Objects* and *Interactions* is called the SOM. SOMs of all *Federates* together constitute the *Federation Object Model (FOM)*.

1.3 HLA Communication Policy

The relations between federation components in HLA framework obey following rules [4]:

1. Federations shall have an HLA Federation Object Model (FOM), standardized in accord with the HLA Object Model Template (OMT).

2. During a federation execution, all interaction of FOM data among federates shall occur via the RTI.

4. Federates shall interact with the runtime infrastructure (RTI) in accord with the HLA interface specification, during a federation execution.

5. Federates shall have an HLA Simulation Object Model (SOM), standardized in accord with the HLA Object Model Template (OMT).

6. Federates shall be able to send and/or receive SOM object interactions externally, as specified in their SOM.

International Journal of Database Theory and Application
Vol.7, No.5 (2014), pp.187-202
http://dx.doi.org/10.14257/ijdta.2014.7.5.14

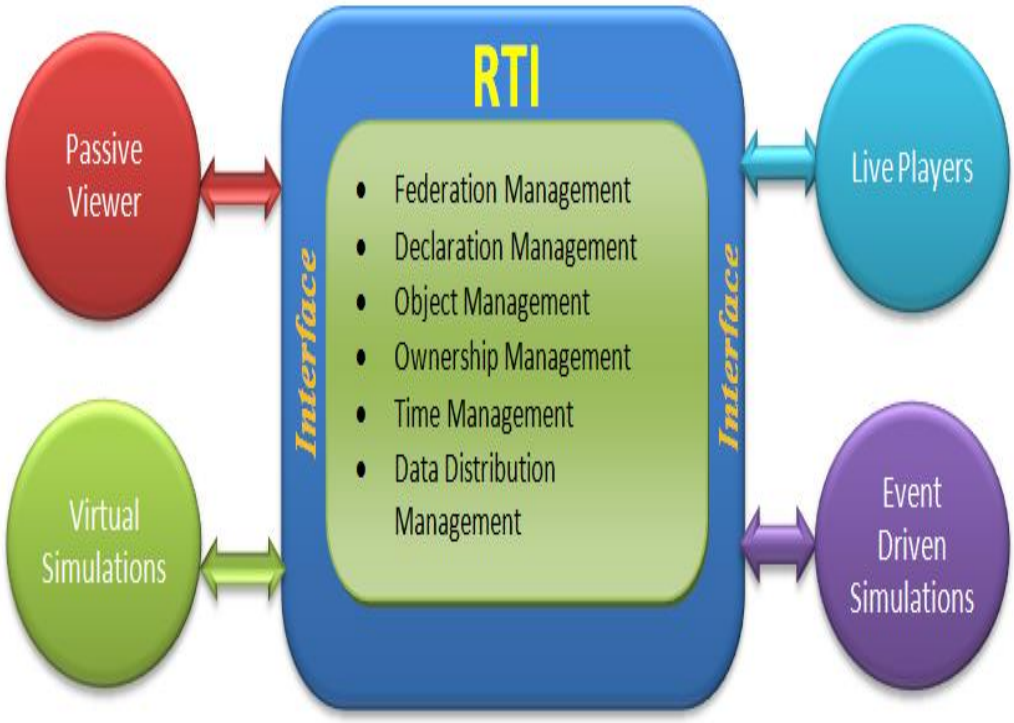

Figure 1. Functional Specification of HLA Framework

HLA framework support federate execution, simulation interaction and interoperability for data exchange between real-time simulations, paced systems, time-stepped simulation and event-driven systems as depicted in the Figure 1.

2. Object Service Object Service Tier (OST) in HLA Framework

Behavioral changes are an important phenomenon during simulation as entities are evolved during course of a simulation. Object Service Tier (OST) is proposed in view of state full and stateless behavioral changes in simulation systems. Object oriented principles, concepts and constructs are applied to achieve the behavioral changes of entities involved in simulation. These behavioral changes are either due to simulation or user intervention. For example, an armored vehicle unit may become a scout and later an artillery observation unit [5]. OST acting a middleware will simplify the use of HLA by providing abstraction and wrapper layer which ease the limitations faced by novice developers. A number of such intermediate layer implementations for the HLA have been available.

2.1 HLA Architecture without OST

An application not introducing an Object Service Tier (OST) will need to call the RTI using the typical services in the HLA interface Specification as described in the previous section "Introduction to HLA Framework"

2.2 Process without Object Service Tier (OST)

Without OST, the application (program/client/federate) has to initiate a remote interface to which it makes calls. Callbacks from the RTI are sent to a federate ambassador object (FAO) that was initially supplied by the application (program/client/federate) [6].
The developer needs to understand at least the core calling interfaces, which are by definition

- Federation Creation
- Federation Joining
- Publishing and Subscribing
- Object Registration

All the development involving code for federate development and interactions with RTI through the interfaces is to be done from beginning.

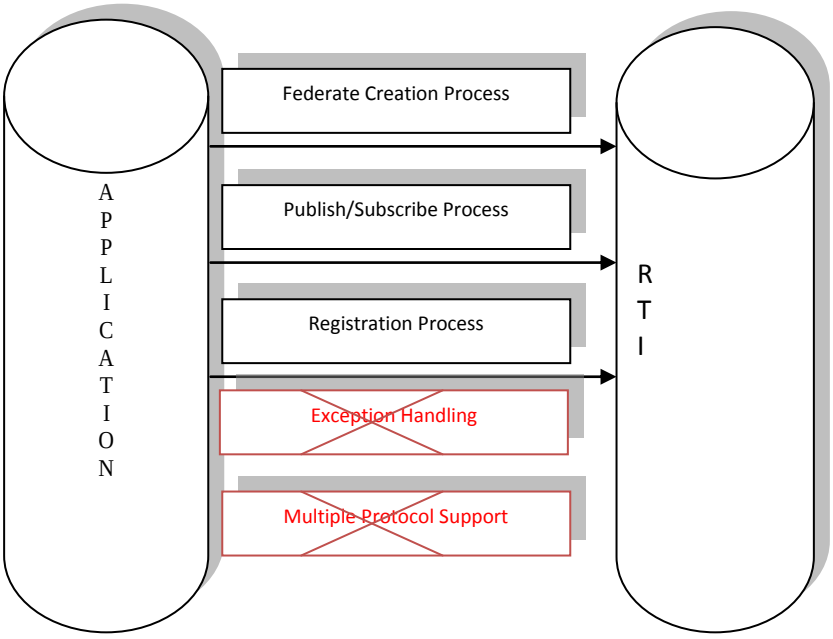

Figure 2. HLA Integration without Intermediate Layer

This indulges all developers to develop an initial prototype of federates with a sample federate and extend it to fit their needs. This process takes time and needs expertise at each level of federate design and implementation. The resulting code will be static and less scalable.

2.3 Process with Object Service Tier (OST)

Introduction of an object-service tier (OST) is designed to yield a level of abstraction between the simulation modeler and the details of the RTI. This will improve development time as well as add scalability and robustness as only fine details are involved [7].

Customary ways of federate execution requires in depth understanding and detail handling of all the federate creation, joining, publishing and registering processes. Proposed object service tier (OST) provides openness, flexibility, and scalability that are needed by enterprises to develop and deploy simulation applications efficiently and independent. Decreased coupling isolates the general requirements and limitation of HLA process from problem design.

2.4 Design Methodology

A basic object service tier OST middleware is shown in the Figure 3 and it consists of: [8]

- An Object Service Tier Core (OST)
- Layer Containers that run within the object service tier (OST) Core (LC)
- Service Modules that run within Layer Containers (SMC)
- Finally federates in the federation and RTI to provide runtime services

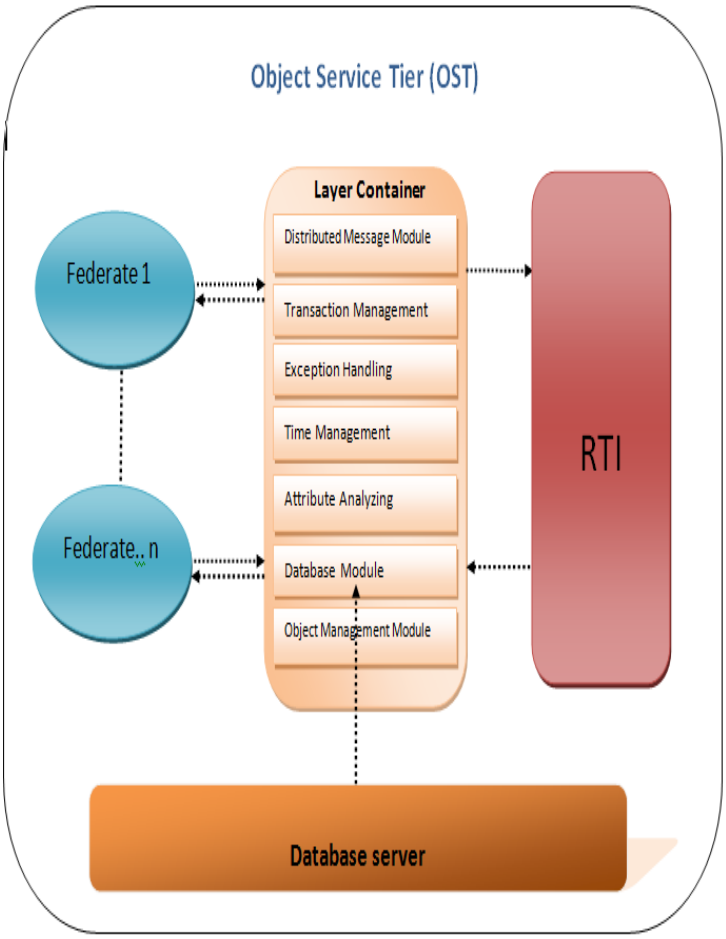

Figure 3. Object Service Tier Architecture

OBJECT SERVICE TIER CORE:

The Object Service Tier (OST) core provides an organized framework and execution environment for layer containers (LC) which encapsulates set of HLA services to run in. OST makes available system services for exception handling, network distribution of messages, multiprocessing, load-balancing among federates, or/and device access for Layer Container

(LC). The Object Service Tier (OST) may also offer control access and permissions for Layer container (LC) services, through creational patterns.

The Object Service Tier (OST) may also facilities simulation and requirement specific features like data optimization, control interfaces, or any additional Services as needed by distributed simulation requirement.

LAYER CONTAINER:

A Layer container (LC) acts as the interface between a Federate and low-level, platform-specific functionality that supports the integration between Federate and RTI. The Layer container is an abstraction that manages one or more core services required by RTI to communicate with federate. Standard interfaces are described in Layer Container (LC) as defined in HLA specification, to provide core services. A federate never accesses a concept/entity or passes a message directly to RTI. Any message or concept is accessed through Layer Container (LC) modules, which in turn invoke the RTI.
There are two types of service methodologies for layer containers (LC):

- *Calling layer containers* that may contain transient, non-persistent services/messages for federates whose states are not saved at all.
- *Entity services layer containers* that contain persistent services/messages for federates whose states are saved between subsequent requests.

So layer container manages different layers of discrete importance to RTI by providing wrapper for multiple services in OST, providing core functionality to RTI and federates.

Two of major HLA architecture constructs are taken into consideration for proposal of design methodology using OST middleware in context of Layer Container (LC) and functional services [6].

3. OOP-FOM Model in Object Service Tier

Federation Object Models (FOMs) are responsible for data exchange within federation (*i.e.*, multiple simulations) [9].The FOM describes a static design that restricts the simulation product dynamism and intricate development efficiency (*e.g.*, it is difficult to extend and structure FOM dynamically after creating a federation).

In HLA specification two types of data exchange models are:[10]

- Federation Object Model (FOM)
- Simulation Object Model (SOM).

Object Model Template (OMT) is proposed to achieve following considerations in the HLA architecture [13]:

(1) It provides systematic blueprint to specify the format of data exchanged at runtime for HLA federations

(2) Standardizes format for specifying the external interface of each HLA federates.

As HLA framework is not strictly object oriented, it imposes entity approach for communication between federates in federation. The information is usually transferred using standard input files. Implementation of such files is difficult static, which requires all effort at design time.

In view of above and other limitations a signature based representation of elements is proposed that arrange the FOM/SOM in the Object Service Tier (OST) middleware for descriptions of concepts/entities to carry out requirements described by object model template (OMT) [14]. The design and structure of projected Object oriented federate object (OOP-FOM) model is depicted in Figure 4.

A signature primarily describes the structure of group of related FOM entities [16]. Including this a signature may also define entity types, their attributes, the types of those attributes, and the relationships allowed between the FOM/SOM entities. In other words, signature provides statements which include a list of goals and a definitive enumeration of key concepts within a domain addressed by FOM/SOM. The Object Service Tier (OST) middleware interprets the information provided in the form of signatures and makes decisions by services managed by Layer Container (LC) without adding or modifying code. Using this multiple FOMs are supported without code modification and across multiple federates in federation.

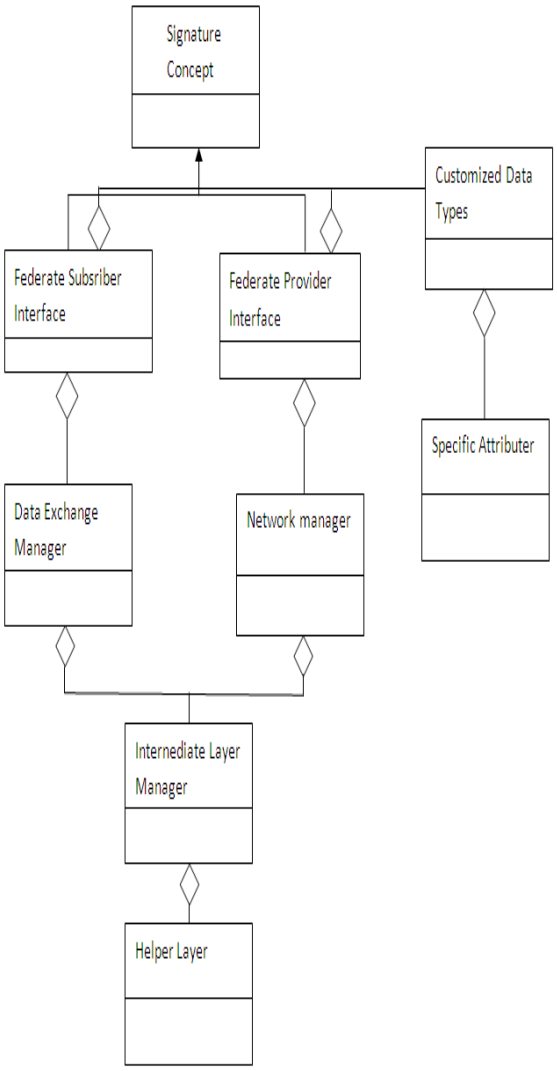

Figure 4. Design Methodology Of OOP-FOM

The signature representing entities involved in a particular FOM can be modeled using:

- Conceptual model,
- Dynamic model,
- Functional model.

The conceptual model represents the artifacts (pieces) of the signature *i.e.*, entities. The dynamic model represents the interaction between these artifacts represented as event, status, and evolution during simulation. The functional model represents the scheme of information exchange flow between participating artifacts.

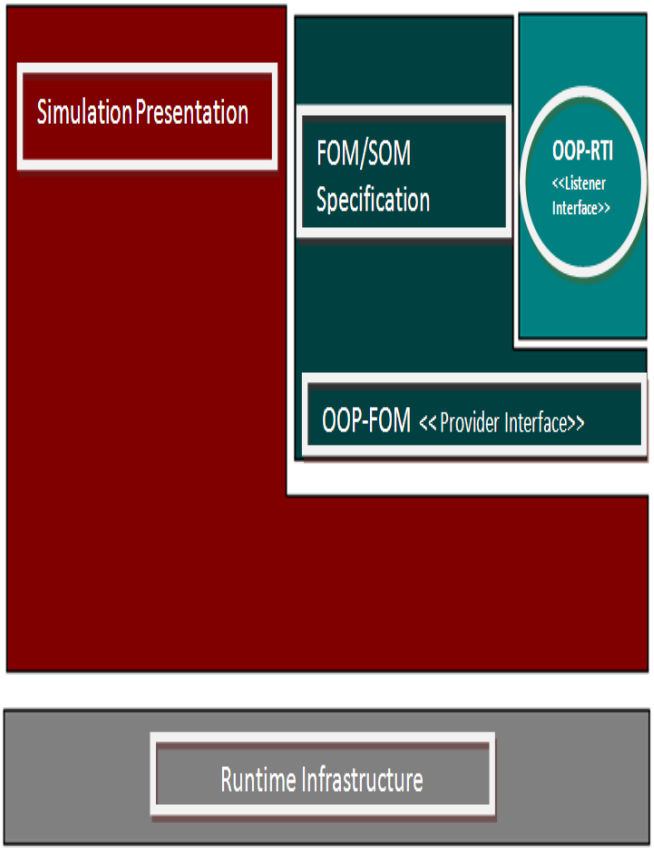

Figure 5. OOP-FOM Systems Flow

As base signature entities will be located at a meta-level with regard to individual concepts and relations between federates, all the related concepts of federate are instances of a signature, as depicted in Figure 5.

Concepts involved in messages may have properties that are complex, and therefore, difficult to generalize. However, in proposed object service tier (OST) the number of possible types (structure) for such properties is predetermined as each federate has a certain role in federation. Many properties of different concepts involve in federation are related through various associations, which adds dynamism by identifying specific OOP-FOM during execution. OOP-FOM define base signature concept that can monitor and modify itself according to a desired goal to support tracking or modification during message passing at runtime.

Properties related to specific entity can be stored as a particular data or structure. This maintains state of a particular concept involved in federation, which helps to describe and identify a certain federation.

4. OOP-Communication Layer in Object Service Tier (OST):

The intent to propose a design methodology of OOP-COMM layer within OST middleware is reusability, data compatibility and providing an abstraction for the distributed separated simulation systems.

OOP-COMM divides Object Model Template (OMT) into discrete parts that can be expanded to form the generalized data communication model. OOP-COMM also assists in dealing with real time decision or application integration problems in an extremely dynamic and agile sphere, such as net-centric war-game. The unique idea of dynamic rapid integration and simulation on demand is achieved efficiently.

4.1 Importance of OOP-Comm Model in OST

OOP-COMM services, managed through Layer Container (LC) in OST middleware will serve as mature commercial off the shelf components.Protocols involved will increase the interopartibilty[17]. The OOP-COMM layer in OST middleware uses interfaces to provide contractual bindings in federate and RTI conforming to HLA interface specification. The interfaces defined by object service tier (OST) isolates the concerns of federates and RTI.

The OOP-COMM model is scalable as the identical interface chain is used by several fedrates in the federation at a time.Customary HLA communication is strongly coupled because of reliance upon static SOM and FOM. OOP-COMM achieve loose coupling due to indirect addressing by involvement of association amoung concern entities involved in data transfer and message passing.Use of interfaces (indirect addressing) provide independence from dissimilar contexts making data interchange model more flexible.

Lagacy HLA systems are based upon bidirectional callback communications between federates and RTI using TCP and UDP for data transportation. RTI has to use many LRS (local resource services) for data exchange while each could connect one federate at the time [17].

In OOP-COMM layer unidirectional request/response interactions and messages are transfered by signatures over data channel reducing network traffic and can address multiple federates. Data encoding used normally in lagacy HLA systems is binary supported while data encoding in OOP-Communication layer are reuseable components based upon readable strings or xml formats which provides paltform and components independency.

OOP-COMM Layer in OST framework is flexible enough to introduce standardized representations of 64 bit data structure/type to be compatible with 64-bit architecture.

As simulation entities change their states and behavior over time, native legacy methodology uses Local Resource Services (LRS) to maintain the states of local federates. Sessions can be used to maintain the state of remote federates in OOP-COMM due to involvement of signatures which are inherently state full. This provides solution to handle multiple federates through several sessions.

Reusability and flexibility features of object service tier (OST) middleware enhance object oriented communication model (OOP-COMM) in its mechanism [14].

Connection between federate may break down due to federate or network failure. Object service tier (OST) will maintain the session for some time and if the connection is recovered in allowed period, simulation will resume which endorse fault-tolerance of simulation federation.

In the meantime, object service tier (OST) checks sessions at regular times. The time-out sessions are treated as invalid ones and will be terminated [16].

4.2 Proposed Design Methodology for OOP-Communication Layer

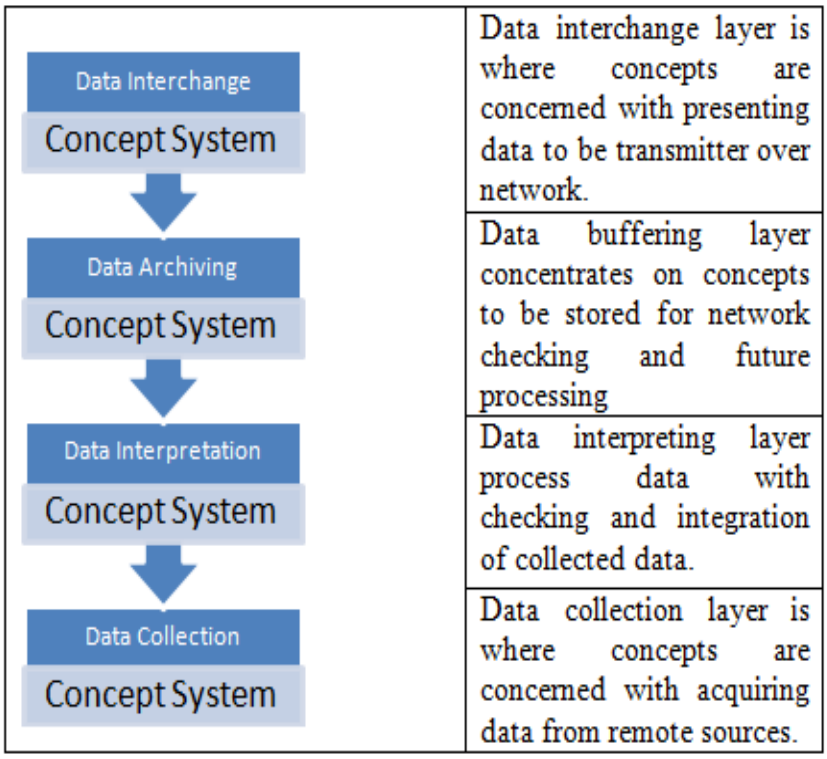

Figure 6. OOP-COMM Layer Processes [21]

The design considerations of OOP-COMM model is described below.

4.3 Design Goals

The OOP -COMM model goal is to achieve the facility which easily replaces any simulation component with little or no software modifications. The OOP-COMM model components are designed to be self contained and provide communication architecture to HLA framework using well defined interfaces. Following are the basic constructs involved to achieve OOP-COMM desired design.

BASE FEDERATE CONCEPT: It is responsible for maintaining the state information, sending and receiving this information over HLA network using OOP-COMM subsystems (federates) request to subscribe/publish entities, messages or interactions. Once registered to HLA network, federates define an interface for update to be notified by OOP-COMM layer.

OOP-COMM layer handles the data that can be transmitted as concepts or interactions. A concept can contain data as attributes and properties. The fundamental difference between concepts and interactions is that the state of a concept persists throughout time, while an interaction is a onetime message that is sent and then ceases to exist.

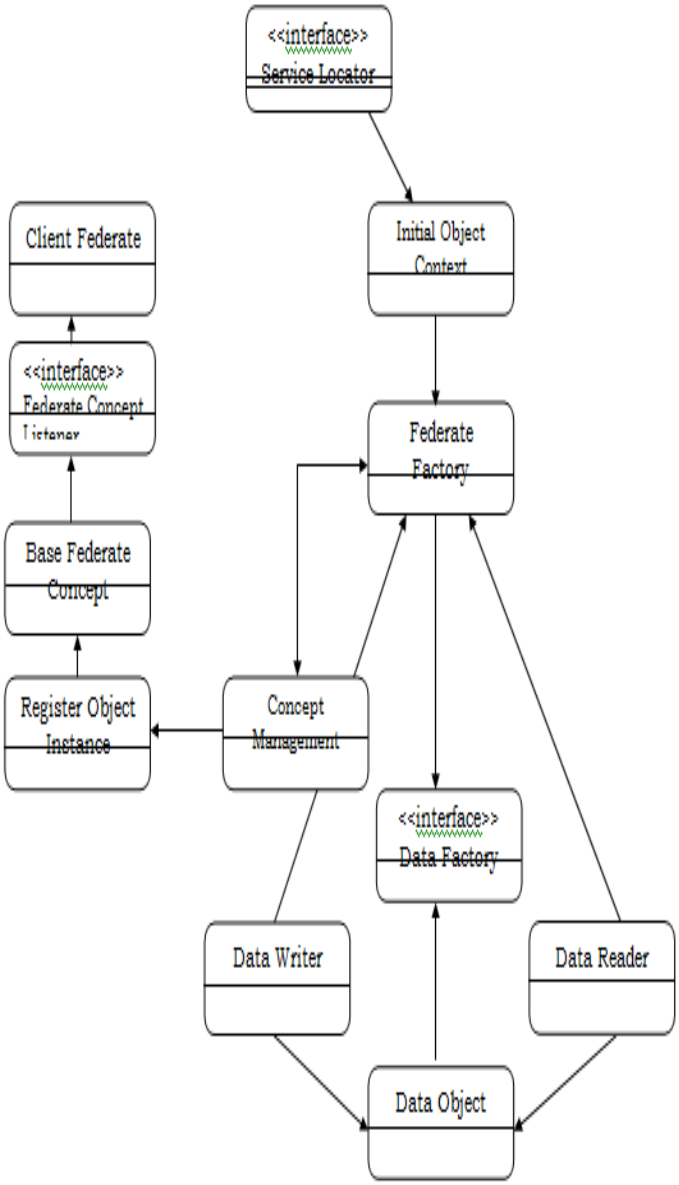

Figure 7. Design Methodology of OOP-COMM

FEDERATE CONCEPT: Federate Factory is responsible for locating, creating and removing instances of base federate concept represented through signature model as depicted in figure above.

FEDERATE CONCEPTS LISTENER: The Federate Concept Listener interface provides methods/services to be implemented by base federate concept. This type of framework makes it possible to create a standard startup template for novice users.

FEDERATE FACTORY: Federate factory provides an abstract layer hiding underlying complexity of resource allocation, thread acquisition involved in instance creation to facilitate calling modules limiting dependency and coupling.

THE FEDERATE REMOTE INTERFACE: The remote interface in the federate provides certain services and lists the service methods available by the federate. The base federate concept is the client's view (simulation) request involved in simulation and implements the remote interface through federate concept listener.

OBJECT SERVICE TIER (OST) INTERACTION FRAMEWORK: The base federate concept itself is contained within an object service tier (OST) container, and should never be directly accessed by external entity or interface. The object service tier (OST) should mediate all base federate concept accesses. The remote interface definitions are provided through federate concept listener which prevents unsecure accesses from different federates in federation. It helps to ensure that the messages are addressed by particular federate concept in OST middleware. It also allow for resource pooling so that certain federate concept instances are created and are in pool waiting to fulfill the request.

FEDERATE AND RTI INTERACTION FRAMEWORK: Federate locate the specific federate concept in OST through Initial Object Context (IOC). The OST is responsible to invoke the required services through Object Management Service (OMS). Federates involved in federation only get a reference to base federate concept instance through federation concept listener. When the federate client invokes a method, the Object Management Service (OMS) receives the request and delegates it to the corresponding base federate concept instance while providing any necessary wrapping functionality like Stream helper classes, object management, data encoding etc.
The federate uses the federate factory to create or destroy instances of a base federate concept. Federate uses the federate concept listener instance, to invoke the methods of an actual federate class.

CONCEPTS MANAGEMENT SERVICE: Concept Management Service (CMS) deals with the registration, modification, and request/response of messages as the federation concepts [22] are made available through Service Locator Interface (SLI). Support Service like Get *ClassName, GettRemoteException* etc is also served through Object Management Service (OMS).

5. Conclusion

The proposed object service tier (OST) middleware approach in HLA Federation architecture minimizes the level of HLA and enables them to focus the simulation behavior of the system [23]. The programming complexities of the RTI are encapsulated and requirements necessary to develop federate software are lessened. Robust conceptual code skeleton for federates and HLA entities is achieved. At the same time extending inherit features of HLA scalability, time management, synchronization, states maintenance and performance.

The Object Service Tier (OST) middleware in HLA reflects the idea of simulation as object model [55]. Using object oriented federate object model (OOP-FOM) layer in object service tier (OST) middleware ensures HLA framework to center on interoperability, interactions between simulation components and effective information interchange among resources [24] by using signature model services of generalization and abstraction for scalability, state maintenance and

performance. The combination of HLA and OOP-FOM model can largely extend the capability of simulation frameworks.

OOP-COMM in OST middleware makes deployment and access of simulation application convenient in distant communication involving firewall access and heterogeneous resources.

Conceptually OST middleware framework is a trade-off between performance and modularization. For Example: OOP-COMM model is based on XML with string-encoded data and the encoding/decoding overhead makes transmission efficiency lower than that with binary data.

Simulation applications that are developed using OST proposed object model have important research value and wide application prospect as they encourage the transformation of current simulation resources and the development of new applications. Further research on other services of OST middleware is underway. OOP-FOM and OOP-COMM will provide the basis for further research. Moreover efficiency of current model in existing HLA implementation will be evaluated using Benchmarks.

References

[1]. W. Y. Min and G. Tag, "Design and Implementations of Surrogates for Interoperation of HLA Federations", Kim School of EECS KAIST 373 -1 Kusong - dong, Yusong -guDaejeon, Korea pp. 305-701.
[2]. High-level architecture (simulation) from wikipedia free encyclopedia, en.wikipedia.org/wiki/High-level_architecture_(simulation).
[3]. J. P. Sousa and D. Garlan, "Formal Modeling of the Enterprise JavaBeans ™ Component Integration Framework September 2000 CMU-CS-00-162 ,School of Computer Science Carnegie Mellon University, Pittsburgh, PA 15213 US.
[4]. Dr. Çağatay ÜNDEĞER, Öğretim Görevlisi, "A Distributed Simulation Standard: High Level Architecture (HLA)", Bilkent Üniversitesi Bilgisayar Mühendisliği Bölümü.
[5]. D. Theotokis, A. Sotiropoulou and G. Gyftodimos, "Transparent Modeling Of Objects Evolution ,Department of Computer Science and Technology", School of Science and Technology, University of Peloponnese, GR 22100 Tripolis, Grecce.
[6]. B. Möller and F. Antelius, "Object-Oriented HLA - Does One Size Fit All?", bjorn.moller@pitch.se fredrik.antelius@pitch.se.
[7]. R. Shweta, "Evolution of Middleware Technology and Its Widespread Applications", Department of Computer Science, Ajay Kumar Garg Engineering College,Ghaziabad.
[8]. M. Salehie, S. Li and L. Tahvildari, "Architectural Recovery of JBoss Application Server,Electrical and Computer Engineering,University of Waterloo,Waterloo", Ontario,Technical Report UW-ECE#2005-02,February **(2005)**.
[9]. W. G. Wang, Y. P. Xu, X. Chen, Q. Li and W. P. Wang, "High level architecture evolved modular federation object model", College of Information System and Management, National University of Defense Technology, Changsha 410073, P. R. China **(2008)**.
[10]. J. S. Dahmann, R. M. Fujimoto and R. M. Weatherly, "The DOD High Level Architecture: An Update", Defense Modeling and Simulation Office N. Beauregard Street Alexandria, VA 22311 College of Computing, Georgia Institute of Technology, Atlanta, GA 30332-0280, The MITRE Corporation,7525 Colshire Drive,McLean, VA 22102-3481, **(1901)**.
[11]. R. Singh and H. S. Sarjoughian, "Software Architecture for Object-Oriented Simulation Modeling and Simulation Environments: Case Study and Approach Technical Report TR-03-09 Sept. Dept. of Computer Science & Engineering Ira A. Fulton School of Engineering Arizona State University Tempe", Arizona, 85287-5406, USA, **(2003)**.
[12]. M. D. Myjak, S. T. Sharp, T. Lake and K. Briggs, "Object Transfer In HLA,The Virtual Workshop, & HLA Products, Inc", P. O. Box 98 Titusville, FL 32781.
[13]. R. R. Lutz, "Migrating the HLA OMT to an IEEE Standard, Johns Hopkins APL Technical Digest, vol. 21, no. 3, **(2000)**.
[14]. Dr. A. Tolk, "HLA-OMT versus Traditional Data and Object Modeling (Updated and Extended Version for the 6. ICCRTS)", Industrieanlagen-Betriebsgesellschaft mbH (IABG) ,Einsteinstr. 20 D-85521 Ottobrunn, Germany.
[15]. The HLA Tutorial, "A Practical Guide for developing Distributed Simulations www.pitch.se/hlatutorial Copyright Pitch Technologies AB", Sweden, **(2012)**.

[16]. B. B. Kristenseny and O. Vilmannzy, "HLA and Simulation Frameworks_ The Maersk Mc-Kinney Moller Institute for Production Technology", University of Southern Denmark/Odense University, DK 5230 Odense M, Denmark.
[17]. O. Topçu and H. Og˘uztüzün, "Multi-Layered Simulation Architecture: A Practical approach", Computer and Information Sciences II, **(2012)**, pp 439-443.
[18]. M. D. Myjak, "Java Real-Time RTI", The Virtual Workshop P. O. Box 98 Titusville, FL 32781 Sean T. Sharp The Virtual Workshop P. O. Box 98Titusville, FL 32781.
[19]. M. D. Myjak, S. T. Sharp and W. Wennie Shu, "PhD, Implementing Object Transfer In the HLA", The Virtual Workshop P. O. Box 98 Titusville, FL 32781 Jeremy Riehl, Demarron Berkley, Phuoc Nguyen, Sean Camplin, Mike Roche,Lockheed-Martin, 12506 Lake Underhill Road, Orlando, FL 32825-5002.
[20]. M. Eklöf, "Fault-Tolerance in HLA-Based Distributed Simulations, Department of Electronic", Computer & Software Systems, TRITA-ICT/ECS AVH 06:03 ,ISSN 1653-6363 ,ISRN KTH/ICT/ECS AVH-06/03--SE © Martin Eklöf, **(2006)**.
[21]. M. Wurpts and R. Logan, "HLA inside and out: Intra- and Inter-Vehicle Communications", Southwest Research Institute, San Antonio, Texas.
[22]. D. Çetinkaya and H. Oğuztüzün, "A Metamodel for the HLA Object Model", Department of Computer Engineering ,Middle East Technical University ,Inonu Bulvari, 06531, Ankara, Turkey.
[23]. K. Cox, "A Framework-based Approach to HLA Federate Development", John Hopkins University / Applied Physics Laboratory, Johns Hopkins Road, Laurel, Maryland 20723.
[24]. W. G. Wang, W. G. Yu, Q. Li, W. P. Wang and X. C. Liu, "Service-Oriented High Level Architecture, European Simulation Interoperability Workshop., Edinburgh, Scotland: Simulation Interoperability Standards Organization. 08E-SIW-022, **(2008)**.
[25]. M. Salehie, S. Li and L. Tahvildari, "A Metric-Based Heuristic Framework to Detect Object-Oriented Design Flaws", Department of Electrical and Computer Engineering, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G.

