Development of an Encapsulator for Interoperability of Non-HLA Based MATLAB Programs (SIMULINK or M-file) to HLA Based Simulations

Sangha Choi, Wooshik Kim, and Sugjoon Yoon

Abstract—MATLAB has been widely used for providing a variety of functions and tools in engineering computing, numerical analysis, modeling and visualization. In particular, through SIMULINK, MATLAB has advantages for M&S (modeling & simulation). HLA is general M&S standard for interoperability and reusability. There is no program that fully supports for linking non-HLA based MATLAB programs into HLA based simulation. In order to build multi-domain interactive simulation system, the encapsulation program for linking non-HLA based MATLAB programs into HLA based simulation is in need of study. In this paper, we propose to structure an encapsulator for interoperability of non-HLA based MATLAB programs (SIMULINK or M-file) to HLA based Simulation. We introduce basic design and implement the encapsulator and show the experiment result in which the encapsulator links Non-HLA based MATLAB programs into HLA based simulation.

Index Terms—HLA, RTI, encapsulator, simulation.

I. INTRODUCTION

Traditionally, MATLAB programmers develop programs using m-file. Since development of SIMULINK providing auto-generated code in M&S, many simulations and developments have been developed by SIMULINK. M&S engineers have proved that SIMULINK is suitable and can be used as universal platforms for engineering analysis and design problems in the scientific and technical development environments.

High-level architecture (HLA) is a standard for building a simulation for a larger purpose by combining several simulations in distributed simulation. This standard supports interoperability and reusability by suggesting a common architecture [1]-[3]. The US Department of Defense (DoD) and others developed standards in the 90s’. These standards were later transitioned to become open international IEEE standards [1]-[3]. The North Atlantic Treaty Organization (NATO) also recommends the standards [4].

HLA has been used in many distributed simulation applications. The HLA provides a set of Application Programmer’s Interface (API) standards to define the architecture [5]. Simulation applications communicate by invoking the HLA API. Run-time Infrastructure (RTI) is a software that implements the HLA API and transfers data from one federation to another. RTI is a fundamental component of HLA, it also implements the interface specification and represents one of the most tangible products of the HLA [5].

HLA is generally difficult to fully understand due to high entrance barriers. In order to build distributed simulations to meet the requirements of HLA standard, M&S engineers have to spend much time to understand the HLA standards to develop programs. In particular to interface RTI with Matlab programs, we need to develop a program with a special function, which we call an encapsulator, for linking non-based HLA program to HLA based simulations. By providing MATLAB/GUI interfaces to encapsulate non-HLA based MATLAB programs, the programs can link into HLA/RTI. In this paper, we develop an encapsulator that fully supports the HLA/RTI interfaces.

This paper is organized as follows: In Section II, we briefly explain related work. In Section III, we describe the Basic design of the encapsulator and present implementation in Section IV. Therefore, In Section V, we show experiment of the encapsulator, and we discuss conclusion and future work in Section VI.

II. RELATED WORKS

Many approaches have been studied, developed, and proven to link MATLAB programs into HLA/RTI.

Sun and Tang showed solving analysis and design problems in the field of weapon systems, and it is usually not possible by conventional programming using high-level languages such as C++ [6]. They implemented a method of HLA/RTI simulation in SIMULINK.

In [7], Xiong et al. have presented MATLAB has several toolboxes serving for different fields as well as SIMULINK system simulation tools. They have introduced a method of making MATLAB programs join into HLA/RTI seamlessly. This method elaborates the principles and designs of HLA-MEX, HLA-MAPI and MATLAB federate framework [7].

ForwardSIM developed ‘HLA Toolbox’, which enables to run HLA federates from MATLAB® and Simulink® [8]. This toolbox integrates different RTIs such that different
versions of the HLA standards can be used without any change to models.

Both [6] and [7] present methods to integrate MATLAB programs with HLA/RTI. These methods require HLA/RTI expertise. In [8], we recognize the first development of MATLAB program with HLA / RTI. The HLA Toolbox requires knowledge of its syntax. In order to support interoperability of non-HLA based MATLAB programs to HLA based simulations for non-expert of HLA, we introduce the encapsulator.

III. BASIC DESIGN

In Fig. 1, we presented a basic diagram of HLA-based simulation. In the HLA-based simulation, the basic unit of simulation is federation. In each federation, there are more than one federates which are linked to RTI (Runtime Infrastructure) with a bus topology. As we can see in this figure, the topology for communication is a bus. In other words, if a federate wants to communicate with each other via RTI (Runtime infrastructure), they should follow some kinds of protocol. For each federate to follow these protocols, everybody involved in HLA-based simulation needs to know these protocols. However, if there is a special program, then we can avoid this kind of problem such that we can have more flexibility to develop simulations with various functions and chance for programmers who do not know HLA/RTI to join these simulations. This we call an encapsulator and this is the purpose of this research.

The essential part of HLA/RTI are well described in the IEEE standards [1]-[3]. The HLA standards consist of the three parts:

1) IEEE std 1516-2010 Framework and Rules [2]
2) IEEE std 1516.1-2010 Federate Interface Specification [1]
3) IEEE std 1516.2-2010 Object Model Template (OMT) Specification [3]

The Framework and Rules specifies ten basic architectural rules which the entire federation should follow [2]. The Federate Interface Specification specifies the interfaces provided by RTI [1]. The Object Model Template (OMT) Specification specifies the HLA object models, such as FOM [3]. The encapsulator also need to follow these standards such that its structure is divided into three management parts to effectively implement the three parts of the HLA standard, which we presented in Fig. 2. As we can see in this figure, the encapsulator is composed of 3 parts, i.e., 1. OMT management part, 2. Interface management part, and finally 3. Rule management part.

A. OMT Management

The OMT management part in Fig. 2 manages a fed file (Federation Object Model Document Data in HLA 1.3), MATLAB files (m-file and SIMULINK), and also presents a buffer for implementation of HLA OMT. This part analyzes all the attributes and parameters in the fed file. The attributes and parameters are picked to connect with value through the buffer. This management part executes the MATLAB files and stores values into the buffer. The values are picked before simulation run. It also matches the attributes and parameters with the values in the buffer.

B. Interface Management

An RTI is a middleware required to implement HLA, which we show in Fig. 3. It provides a set of interfaces for supporting federates to coordinate their operations and data exchange during a simulation. By coding the RTI syntax in the encapsulator, the interface management part manages the RTI interfaces to support the HLA Interface Specification.

Fig. 4 illustrates RTI and federates code responsibilities. The HLA Interface Specification identifies the services provided by libRTI to each Federate and the obligations that each Federate bear to the Federation. Within libRTI, the RTI Ambassador class bundles the services that RTI provides. Every request made by federate in RTI takes the form of an RTI Ambassador method call. The abstract class FederateAmbassador identifies the callback function that
each Federate should provide.

RTI and Federate “Ambassadors”

Fig. 4. RTI and federate code responsibilities.

Fig. 5 shows general process between federate and federation [5]. The process satisfies the HLA interfaces. Some of the important interfaces are listed below:
- Federation management
- Declaration management (DM)
- Object management
- Ownership management
- Time management
- Data distribution management (DMM)
- Support services

The encapsulator follows the HLA interfaces by coding the RTI interfaces in the process with the code responsibilities.

The encapsulator follows the HLA interfaces by coding the RTI interfaces in the process with the code responsibilities.

C. Rules Management

Rules management block encloses the first two management blocks, i.e., the OMT management and Interface management blocks, and manages the entire process of simulation execution to meet the ten architectural rules. These rules are about principles and relationship between federates and the federation with HLA OMT and HLA interfaces [2].

IV. IMPLEMENTATION

In this section, we present the implementation of the encapsulator including the five functional fields, the execution sequence, and the development environment of the encapsulator.

A. Five Functional Fields

Fig. 6 shows the five fields of the encapsulator. These 5 fields are Scenario, Buffer, Connector, Execution file, and the Data field. The Scenario field manages fed files. In the fed files, the scenario of the simulations are summarized. The field visualizes all the components of the fed files and picks all the attributes and parameters automatically for exchanging data via RTI. It provides few functions as buttons for setting attributes and parameters in a federate. The encapsulator also provides

The Execution file field enables to execute MATLAB files and saves values. In this field, users can check, edit input parameters and output parameters. It also provides a setting of initial values of the input parameters.

The Buffer field manages a buffer. In this field, users can check the attributes and parameters by setting values and types of data.

In the Connector field, users can do Federation managements such as creating federation and federates, joining federates into the federation, resigning a federate, or removing the federation.

In the Data field, we can see the every exchange of data and such that we can analyze the simulation.

B. Execution Sequence

In Fig. 7, we showed a sequence diagram of the encapsulator, which are composed of 4 parts. These parts are 1. Data classification part, 2. Managing HLA Interfaces part, 3. Data Exchange part, and finally 4. End Simulation part. Short explanation of each part are as follows;

1) Data classification part: The encapsulator requires a fed file and a MATLAB file. Users can pick visualized attributes and parameters what the federate subscribes and publishes from the fed file. The User also can set initial input on the MATLAB files.

2) Managing HLA Interfaces part: The next sequence of the classifications is connection to RTI. The encapsulator creates a federate to join into the federation that which the users named (if there is no the federation, it creates a new federation). The encapsulator manages the HLA interfaces via the RTI after the connection.

3) Data Exchange part: The encapsulator runs the
MATLAB files. It puts the initial input parameters and saves output parameters in a buffer. The buffer is a name table to link the attributes and the parameters to value of the MATLAB program. The encapsulator exchanges the data what the users set in the part 1.

4) End Simulation part: When the end condition is reached, the encapsulator resign the federate and the federation.

C. Development Environment

In Table I, we summarized development environments. The encapsulator is a MATLAB-based encapsulation program with many advantages in M&S engineering.

| OS      | Tools                                  | Language |
|---------|----------------------------------------|----------|
| Window 10 | MATLAB R2017a, MAK RTI 4.4, MatlabHLA-Toolbox | MATLAB   |

In Fig. 8, we presented a MATLAB/GUI interfaces to link MATLAB programs to HLA/RTI [8], [9]. In particular, as providing the Scenario and Execution field, it makes it significantly easier to link MATLAB programs into HLA/RTI. In addition, the Data field shows every data exchange to analyze simulation data.

![Fig. 7. Encapsulator sequence diagram](image)

![Fig. 8. Structure of MatlabHLA-Toolbox [7].](image)

Generally, RTIs support APIs to C++ and/or Java conform to the Interface Specification of the HLA standard. As illustrated in Fig. 8, we have developed the encapsulator with the MatlabHLA-Toolbox [9] that wraps m-file API to C++ API of RTI and MALTAB External (MEX) interface which allows access to external libraries such as C++. It allows the encapsulator to be developed without the C++ code.

There are different commercial and non-commercial RTIs. Among these, we used MAK RTI 4.4 of VT MAK [10] to connect the encapsulator to RTI and support HLA 1.3.

![Fig. 9. Components of experiment federation and federates.](image)

**V. Experiment**

In this Section, to check the performance of the encapsulator developed we performed an experiment. The experiment is based on a recommended practice by IEEE for the HLA Federation Development and Execution Process (FEDEP) [11]. In this experiment, the scenario is as follows; 1. an aircraft moves forward to one direction, 2. a radar station is located on the ground where the aircraft will pass away, 3. The aircraft soon enter into a detecting range of the radar station, and 4. The radar station launches a missile when aircraft comes into the detection area, 5. The launched missile flies to the aircraft and as soon as the missile enters into the predefined striking distance, the missile explodes with the aircraft as well. If the missile do not enter into the striking distance, then missile flies away and the aircraft continues on flying.

| Federation Name | Entity Name | Entity Type | Description               |
|----------------|-------------|-------------|---------------------------|
| Electronic-Warfare | aircraft_lo | Publish     | Coordinates of the aircraft |
|                  | radarstation_lo | Publish    | Coordinates of the radar station |
|                  | missile_lo | Publish     | Coordinates of the missile |
|                  | missile_m | Publish     | Angle of the missile |
|                  | tscale | Publish     | Target radar range of the missile |
|                  | nscale | Publish     | Search radar range of the missile |
|                  | fiscale | Publish     | Explosion Range of the missile |

As shown in Fig. 9 and Table II, we create a federation and two federates. The Federate 1 named “Electronic-Warfare” is programed in SIMULINK for the entire experiment model, it publishes location of the aircraft, radar station, and missile, and ranges. Federate 2 named “3D-viewer” is programed in m-file for watching the simulation in 3D animation, it subscribes every entity of Federate 1.

In order to link non-HLA based MATLAB programs to HLA based simulation, we developed the encapsulator and
tested. It is a program for non-experts of HLA / RTI. As shown in Fig. 10, the encapsulator successfully created the federation and federates, and joined federate into the federation. In addition, as illustrated in Fig. 11 and Fig. 12, the encapsulator executed the SIMULINK file and m-file, exchanged data via the RTI without any problem.

VI. CONCLUSION AND FUTURE WORK

MATLAB has been widely used to provide a variety of functions and tools in engineering computing, numerical analysis, modeling and visualization. Especially, with plentiful of functions of SIMULINK including 3D VR, animation, GUI, and libraries of various functions blocks, MATLAB has become easier and easier to handle and become more powerful, and has advantages over other programs for M&S (Modeling & Simulation). HLA is a general M&S standard for interoperability and reusability and become a worldwide standard for all the distributed simulations including M&S for military simulations. There has not been an encapsulation program that fully supports for linking non-HLA based MATLAB programs into HLA based simulation. In order to build the multi-domain interactive simulation system, the encapsulation program for linking non-HLA based MATLAB programs into HLA based simulation needs to be studied. HLA is generally difficult to fully understand due to its high entry barriers. In this paper, we have developed an encapsulator and checked the encapsulator links non-HLA based programs into HLA / RTI. With this result, we hope that this encapsulator will make many MATLAB engineers who do not have HLA expertise to feel ease to develop and to take important parts in HLA/RTI-based simulations. Our future work will involve maintaining the program and updating it for HLA 1516 standard.

CONFLICT OF INTEREST

The authors declare no conflict of interest.

AUTHOR CONTRIBUTIONS

Wooshik Kim designed and revised this research. Wooshik Kim, Sangha Choi conducted the research. Sangha Choi wrote this paper, implemented the program, and performed the experiment. Sugjoon Yoon analyzed the HLA and managed the project. All authors had approved the final version.

REFERENCES

[1] IEEE, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Federate Interface Specification, IEEE Std 1516.1-2000.
[2] IEEE, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) - Framework and Rules, IEEE Std 1516, 2000.
[3] IEEE, IEEE Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) – Object Model Template (OMT), IEEE Std 1516.2-2000.
[4] NATO, STANAG 4603: Modeling and Simulation Architecture Standards for Technical Interoperability: High Level Architecture (HLA), 2015.
[5] RTI 1.3-Next Generation Programmer’s Guide Version 3.2, Department of Defense, Defense Modeling and Simulation Office High Level Architecture Run-Time Infrastructure.
[6] G. T. Sun and S. Tang, “HLA-based engineering level warfare system with interactive modeling environment,” College of Astronautics, Northwestern Polytechnical University, Xi’an 710072.
[7] W. Xiong, P. S. Fan and H. Y. Zhang, “HLA based collaborative simulation with MATLAB seamlessly embedded,” International Journal of Machine Learning and Computing, vol. 2, no. 2, April 2012.
[8] ForwardSim. (March 2018). HLA Toolbox. [Online]. Available: http://www.forwardsim.com/products/hla-toolbox/
[9] C. Stenzel and S. Pawletta, “Matlab O R h i g h l e v e l a r c h i t e c t u r e toolbox,” Computational Engineering and Automation, University of Wismar, February 2008.

Sangha Choi is currently a M.S. student. He studies at the Department of Electronics, Information and Communication Engineering in the University of Sejong, Seoul, Korea. His current research interests are DSP and its application to communications, modeling and simulation, etc.
Wooshik Kim received his B.S. and M.S. degrees in electronics engineering in 1984 and 1986, respectively, from Seoul National University, Seoul, Korea, and his Ph.D. degree in 1991 from Georgia Institute of Technology. Currently, he is a professor with the Department of Information and Communications Engineering, Sejong University. His current research interests are DSP and its application to communications, VHDL, sensor networks, modeling and simulation, etc.

Sugjoon Yoon holds a Ph.D. degree in aerospace engineering from the University of Michigan, USA, 1987. He received his M.A. and B.A. from Seoul National University, Seoul, South Korea, 1983. For past 15 years, he has been working as a professor at Sejong University, South Korea. His research interests include flight dynamics, control, hardware-in-the-loop simulation. He is the CEO of Model Sim. Prof. Sug Joon Yoon is a fellow member of AIAA.