HLA LISTENER – A GUI DRIVEN MAPPING OF HLA STANDARD SERVICES

Mostafa Ali[(a)], Yasser Mohamed[(b)]

[(a)]PhD Candidate, University of Alberta
[(b)]Associate Professor, University of Alberta

[(a)]MostafaAli@ualberta.ca
[(b)]yaly@ualberta.ca

ABSTRACT
High Level Architecture (HLA) is a distributed simulation framework that offers integration, reusability, and flexibility. Nonetheless, it has been criticized for its complexity and overwhelming services. Consequently, it has been rarely used on a day-to-day basis in industrial world. A very steep learning curve is one of the main challenge that faces a new user of HLA distributed simulation. The user has to be familiar with different HLA services (e.g. objects and time management) beside a programming language to develop a basic simulation. In order to overcome this challenge, we developed a tool with friendly graphical user interface that encapsulates all HLA services. This tool allows a new user to explore services as described in the standards without the need for programming language. In addition, the interface can be used by an experienced simulator to debug existing federation quickly and efficiently.

Keywords: Distributed Simulation, HLA, HLA Listener

1. INTRODUCTION

Simulation provides an affordable means for teaching, analyzing, and testing complex physical system that would usually require a huge investment and manpower to perform in a real-world environment. For example, flight simulation has been widely used in commercial and military aviation for training potential pilots (Page, 2000).

Over the years, Different simulation paradigms such as discrete event, agent based, system dynamic simulations have been proposed to address specific domains and applications. Each simulation approach advocates a modelling system that simplifies real-world system. For example, discrete event simulation depends on event scheduling basis to simulate real world scenarios (Goti, 2010). While Agent-based simulation relies on a set of autonomous agents that interacts with each other and with the environment to simulate systems like socio-economic science (Helbing, 2012; Jennings, 2000; Wooldridge, 1997).

On the other hand, distributed simulation takes a unique approach by suggesting decomposing the simulation domain into separate manageable components that interact with each other to provide a complete simulation solution. Distributed simulation manages interactions and interoperation between different simulation components; however, it does not intervene with any of the component internal mechanism. Each component can use different simulation type internally to model its scope without affecting another component. This powerful flexibility allows combining multiple simulation model into one scenario easily and effectively.

High Level Architecture (HLA) is a popular distributed simulation standard. HLA has been originally developed by the Department of Defense in the United States (Borshchev et al., 2002; Fujimoto, 2003). It is currently regulated by IEEE, and the latest version is 1516-2010 known as HLA evolved.

Distributed simulation has been used in different domains; like logistical processes, traffic management systems, and freight transportation (Schulze et al., 1999; Zacharewicz et al., 2011); optimizing supply chain from manufacturing through distributer to end customer (Turner et al., 2000); tunneling (AbouRizk, 2010); fleet optimization in earthmoving operations (Ali et al., 2014), manufacturing (Hibino et al., 2002), 3D virtual environment for marine port environments (Bruzzone and Longo, 2013), and combat simulation (Ham et al., 2014).

Although distributed simulation has many potential beneficial application in industry such as supply chain and digital factories (Taylor et al., 2002), it is not fully embraced in industry comparing to research and military domains (Boer et al., 2006a; Lendermann et al., 2007; Taylor et al., 2002). Poor understanding of industrial needs, complexity of distributed simulation, ambiguities in HLA standards, and steep learning curves are main reasons that hinder applying distributed simulation on day-to-day basis in industrial domain (Boer et al., 2008, 2006b; Taylor et al., 2002).

A non-expert user who wants to investigate HLA distributed simulation has to spend substantial time to gain knowledge in three areas: 1) Distributed simulation; 2) Understanding HLA standards ("IEEE Standard for Modeling and Simulation (M\&S) High Level Architecture (HLA)-- Federate Interface Specification," 2010); and 3) Using a programming language to write a simple start-up distributed simulation scenario. This steep learning curve turns practitioners away from

embracing distributed simulation. This shows the necessity for providing simplified approaches that allow non-expert to develop distributed simulation with minimum knowledge requirements.

This paper shows our effort to develop an intuitive graphical interface that encapsulates all services provided in HLA distributed simulation. Graphical User Interface dramatically improves reception, and learning (Gerhardt-Powals, 1996; Staggers and Kobus, 2000), as it simplifies complex concepts into an intuitive user interface that is familiar to anyone with a minimal computer knowledge.

The developed prototype, named HLA Listener, allows non-experts to try distributed simulation with almost zero-learning-curve. Through the interface, the user can create a new federation, join an existing one, and interacts with other federates without the need for a programing language.

Although HLA Listener is not sufficient to develop a complete real-world scenario simulation, it can be used for two scenarios. First, it provides a first step for learning and understanding basic HLA distributed simulation concepts such as: Federation management, Object management, and Time management. In addition, it can be used by experts to debug existing federation through scripting window as will be shown later.

The remainder of the paper is structured as follows: The next section will discuss the structure of HLA Listener and discuss briefly its different components; then a sample use of HLA Listener in training and debugging is provided. Finally, we will conclude by providing our vision for potential improvements for this tool.

2. HLA LISTENER STRUCTURE

HLA Listener aims to provide an intuitive interface for all HLA distributed services to allow a beginner to make sense of the distributed simulation without requiring knowing much about HLA standards or mastering any programming languages. HLA Listener is written in Java which is one of the two Application Programming Interfaces (API), along C++, that are considered part of the HLA standards. Despite that the HLA standard encourages third-parties to develop APIs for other programming languages, we found that most of the commercial and open-source RTI implementations support these two APIs only. Java Virtual Machine® allows running HLA Listener on different operating systems without having to rewrite the code.

HLA Listener consists of five main components, as shown in Figure 1, **HLA Interface** which is the Java API that ships with HLA standard and it allows running the simulation with any HLA-compatible RTI implementation; **FOM parser** retrieves the simulation FOM and provides object classes, interaction classes, attributes, parameters, and data types to the other components; **RTI ambassador** is responsible for sending requests from the user to the RTI, these requests represent different services provided in the HLA standards such as “create a federation” and “Publish object class”; **Federate ambassador** receives callbacks, such as “Discover object instance” and “Time regulation enabled” from the RTI and displays them to the end user; **Scripting module** allows the automation of RTI and Federate ambassador tasks by writing java scripts as will be shown later.

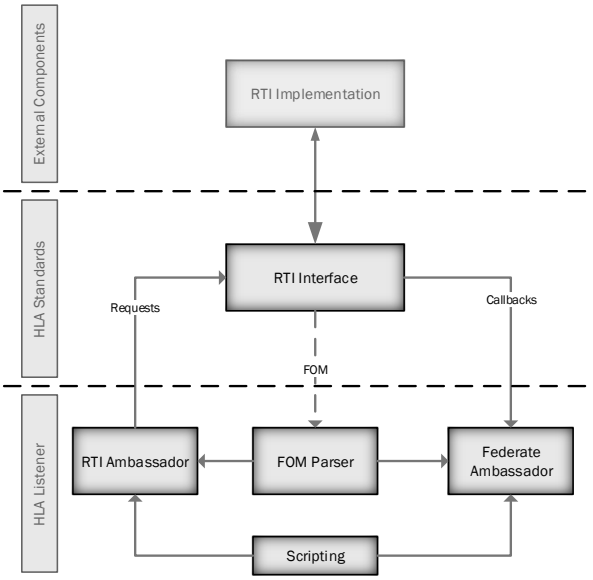

Figure 1 HLA Listener Structure

2.1. Dynamic Link Compatibility

One of the key issues with earlier versions of HLA distributed simulation was the overhead effort required to run the simulation with different RTI implementations; as each RTI implementation has different interface specifications and the simulator has to readjust the simulation to comply with the RTI. The latest version of HLA standards (“IEEE Standard for Modeling and Simulation (M\&S) High Level Architecture (HLA)-- Federate Interface Specification,” 2010) resolves this problem though Dynamic Link Compatibility (Möller et al., 2008). Dynamic Link Compatibility enables the simulator to develop a simulation using a specific RTI implementation that can be replaced with another implementation without requiring any changes in the simulation’s internal structure.

In addition, Dynamic Link Compatibility allows selecting the RTI implementation dynamically at runtime. Using this feature, HLA Listener can work with any HLA-compatible RTI implementation. When launching HLA Listener, it will ask the user to select RTI implementation to run the simulation; we tested it with two commercial and one open-source RTI implementation, and it works as expected without having to change any line of code. The following sample java code demonstrates Dynamic Link Compatibility.

```
public class DynHLA {
    //List all available RTI
implementations
    RtiFactoryFactory.getAvailableRtiFa
ctories().stream().forEach((rti) -> {

    System.out.println(rti.rtiName());
    });
```

```
    private final String preferedRTI =
"RTI A"; // the name of the RTI
implementation
    //Work with a specific RTI
implementation
    RtiFactory rtiFactory =
RtiFactoryFactory.getRtiFactory(prefer
edRTI);
}
```

2.2. FOM Parser

HLA Standard defines Federation Object Model (FOM) "*A specification defining the information exchanged at runtime to achieve a given set of federation objectives. This information includes object classes, object class attributes, interaction classes, interaction parameters, and other relevant information*" ("IEEE Standard for Modeling and Simulation (M\&S) High Level Architecture (HLA)-- Federate Interface Specification," 2010). Latest HLA standards support modular FOMs (Möller et al., 2007) which allows different federates to add one or more FOM files. Those FOMs along MOM and Initialization Module (MIM) are merged into one simulation FOM according to merging rules specified in HLA standards.

This flexibility means that FOM module could change during the simulation model and a well-designed federate should track those changes during simulation execution. HLA Listener tracks any changes in simulation FOM during the simulation execution by subscribing to "HLAcurrentFDD" attribute in "HLAobjectRoot.HLAmanager.HLAfederation" class, which is part of the standard MIM. By subscribing to this attribute, RTI will send a "Reflect attribute value" callback to HLA Listener whenever the simulation FOM gets updated; the callback contains the updated simulation FOM in XML format.

Because HLA Listener is a general-purpose tool and the FOM is processed solely based on the callback provided by the RTI. FOM parser component processes the FOM to get the following information:

1. **Date types**: FOM parser extracts all data types: basic, simple, enumerated, array, fixed record, and variant record. Those data types are used in many HLA services such as "Update attribute value"
2. **Update rate**: FOM module contains different update rate (in Hz) that specifies the maximum rate for sending updates to a specific federate when used with "Best-effort attribute". Update rate is used in many HLA services such as "Subscribe Object Class Attributes".
3. **Transportation type**: There are two basic transportation types "Reliable" and "Best-effort"; however, a FOM might define additional transportation types.
4. **Dimensions**: It defines intervals for each attribute in a format [0, upper value], which is used for Data Distribution Management (DDM) service.
5. **Object Classes and Attributes**: a FOM contains a hierarchy of object classes and their attributes, these classes are used in exchange information between federates. FOM parser retrieves all classes that are used in many services such as "Publish Object classes attribute" and "Register Object Instances" services.
6. **Interaction Classes and Parameters**: Interaction classes and their parameters are very similar to Object Classes and they are used in "Publish Interaction Class" and "Send Interaction" services.

2.3. HLA standard mapping

Although HLA Listener simplifies HLA distributed simulations' services, it does not obfuscate it. The user can still map all graphical components to the standards.

In order to encourage the user to utilize and lookup the standards, each graphical component in HLA Listener has been numbered according to its section number in the standards ("IEEE Standard for Modeling and Simulation (M\&S) High Level Architecture (HLA)-- Federate Interface Specification," 2010). The section number and text in standards are displayed for Menu, Menu items, dialog boxes' titles, requests, and callbacks as in Figure 3.

3. HLA LISTENER EXAMPLE

This section provides an example of using HLA Listener to explore HLA distributed simulation. It does not provide a complete demo of its capability but rather a short introduction of the main steps in a distributed simulation life cycle as shown in Figure 2. An interested reader can come up with more advanced scenario to use HLA Listener to explore advanced services in the HLA standards.

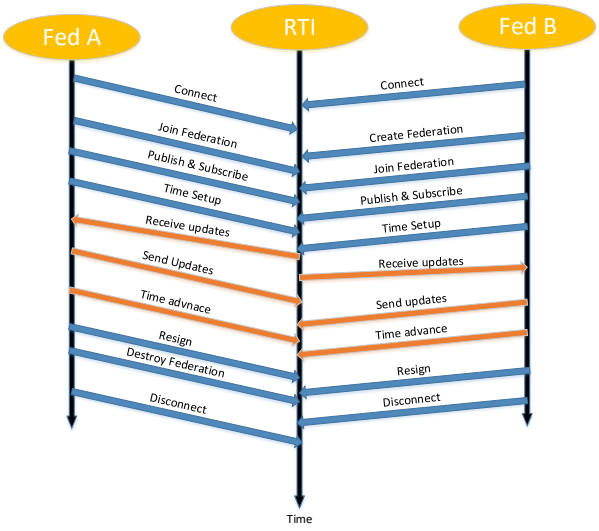

Figure 2 A typical life cycle of HLA distributed simulation.

When a user runs HLA Listener, it will ask for a jar file for the HLA-compliant RTI to connect to through

Dynamic Link Compatibility. All RTI implementations (commercial and open-source) would work exactly the same way. If the link is successful, the main interface for HLA Listener will be shown as in Figure 3. Each menu item represents a section in the HLA distributed simulation standards. The left part shows a history of requests and callbacks while the right section gives more details about the selected request/callback.

The latest version of the standards introduced the concept of the "Connection Mode" which flags each federate either connected or disconnected. Any HLA service, such as create or join federation, must run by a connected federate. Consequently, the first step is connecting to RTI through "Connect service" in Figure 4 (a).

One of the connected federates creates the federation execution by providing its name, a set of FOM modules, and federation logical time type (integer or float) as in Figure 4 (b). Once created, all federates join the federation through the "Join federation execution" as Figure 4 (c). Each federate has the chance to add additional FOM modules during joining the federation.

Time management is a key service in HLA standards; however, a beginner might find it hard to understand all of its associated concepts such as "Look ahead", "Time regulation / constrained", and "Advance in time". HLA Listener can be used to explore all scenarios related to the time management as in Figure 4 (d).

In order to exchange data between federates, each federate should declare object classes and interactions that it will publish or subscribe to. RTI uses these declarations to manage the transfer of updates from publishing federate to subscribing ones. A federate might publish / subscribe many object classes and interactions and an object class / interaction can be published or subscribed by many federates. The declaration menu in HLA Listener can be used to publish / subscribe as in Figure 5 (a). A federate that publishes an object class should register object instance as in Figure 5 (b) and a subscribing federate will receive "Discover Object Instance" callback.

Publishing data in the federation is done through "Update Attribute Values" and "Send Interaction" services, while receiving the data is done though "Reflect Attribute Values" and "Receive Interaction" services. "Update Attribute Values" service, shown in Figure 5 (c), requires providing values for owned published attributes for an instance along a user-supplied tag and optional time stamp.

After sending all updates and processing all received values, a federate should request advance in time through "Time Advance Request" as in Figure 5 (d), "Time advance Request Available", "Next Message Request", "Next Message Request Available" services. Time advance tells the RTI that the federate will not send updates with time stamp less than requested time advance plus look ahead value (Figure 4 (d)). When all regulating federates request to advance a certain point in time, RTI will send "Time Advance Granted" to all federates requesting time advance.

This cycle of sending / receiving updates, request time advance, and time advance granted may be repeated many time during federation execution. When the simulation is complete, each federate should exit gracefully by invoking "Resign Federation Execution" service as in Figure 5 (e), afterwards, one federate should destroy the federation by invoking "Destroy Federation Execution" service as in Figure 5 (f). Finally, all federates should disconnect from the RTI.

The previous demo briefly explained the workflow for a beginner user who wants to explore HLA distributed simulation capabilities without going into too much details. However, HLA Listener can be also used by professionals to test and debug existing federations. If, for example, a federate is expected to receive specific values at certain point of time but it did not, then we have to determine if there is a problem with the sending or receiving federate. Typically, a minimal federate will be used to replace either the sending or receiving federate to determine the problem source. Instead of writing this minimal federate from scratch, HLA Listener can be used for this purpose. A professional user can use the interface to imitate sending or receiving the values. Additionally, the script window, Figure 6, can be used to write a minimal code for testing and debugging.

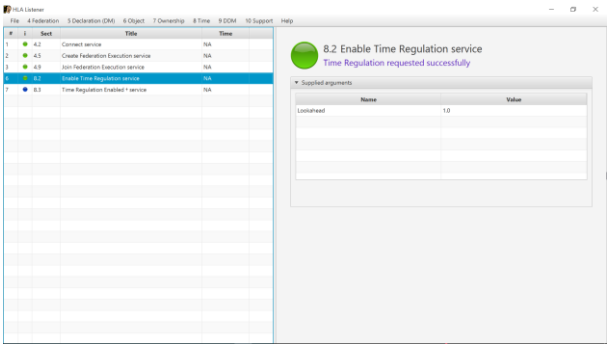

Figure 3 The main interface of HLA Listener.

4. LESSON LEARNED AND FUTURE WORK

The comprehensive investigation of HLA services and its standards during the development of the HLA Listener revealed a number of issues that has to be addressed in the next version of HLA standards. These issues are related to ambiguities and discrepancies in the HLA standards as follows:

1. HLA standard (1516.1 section 4.3) states that "Disconnect service" should raise "Not connected" exception if a disconnected federate invokes "Disconnect service". However, the accompanying Java API does not include this exception.
2. According to the standards, multi-dimensional array is supported. However, it was left to the simulator to determine how to encode and chain different array's dimensions (i.e. by row or column basis) which will severely affect simulation integration and reusability as

different simulators might opt to different encoding preferences.

3. There are not enough learning resources that explain in details the HLA distributed simulation.

For future work, we are planning to add the capability to record user interaction with HLA Listener and auto generate the corresponding code. This will reduce developing time and effort significantly and allows the development of more advanced and complex federation with minimal coding effort. Another proposed enhancement would be the ability to add custom basic data types by manually defining its encoding and decoding factories.

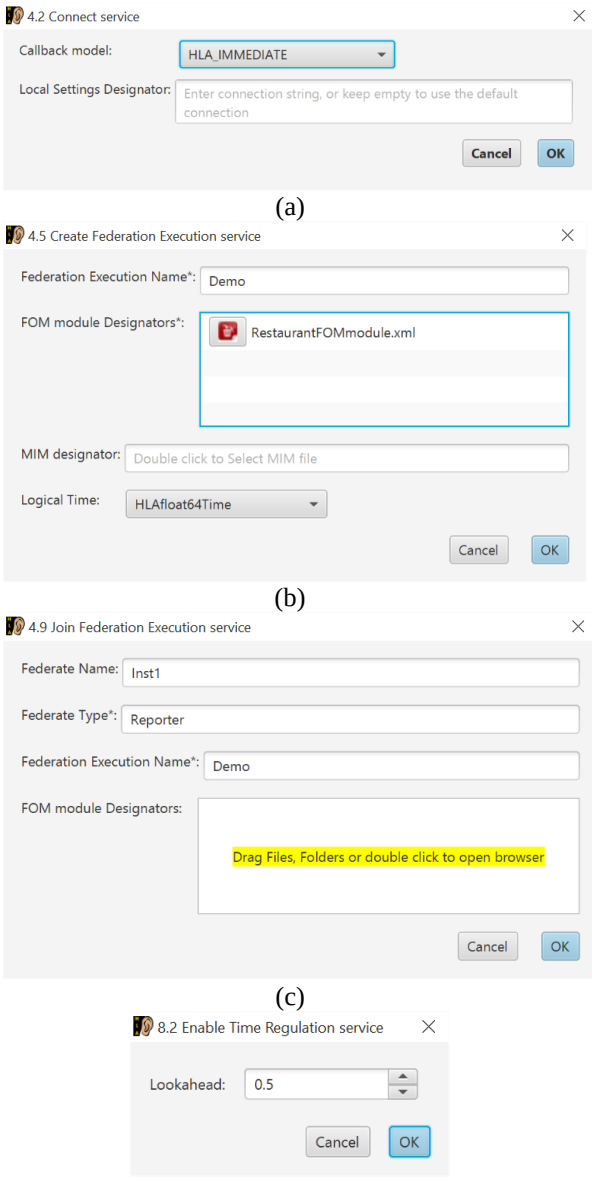

Figure 4 A typical federation execution will start with a) Connect, b) Create Federation (if not exist), c) Join Federation Execution, and d)Enable time regulation / constrained.

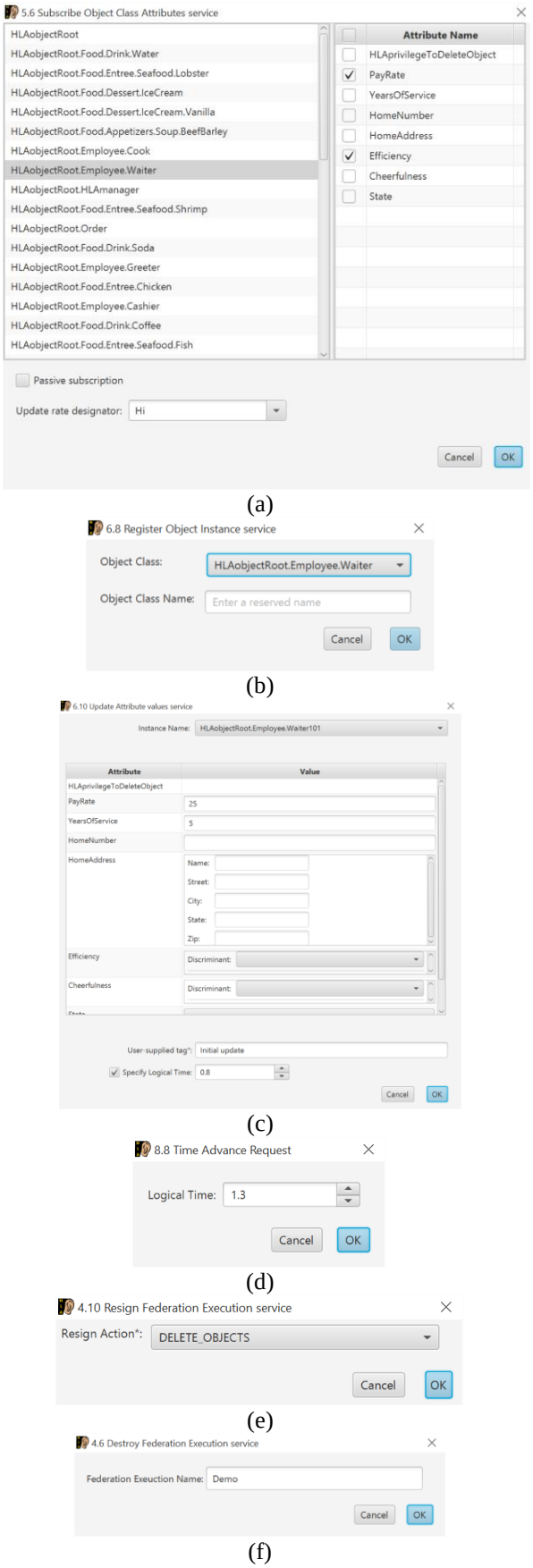

Figure 5 a) Publish / subscribe object classes and interactions, b) Register / discover object instance, c) Send / receive updates, d) Advance in time and repeat

steps c) and d) for the simulation execution lifecycle, then e) Resign federation, f) Destroy federation and / or disconnect.

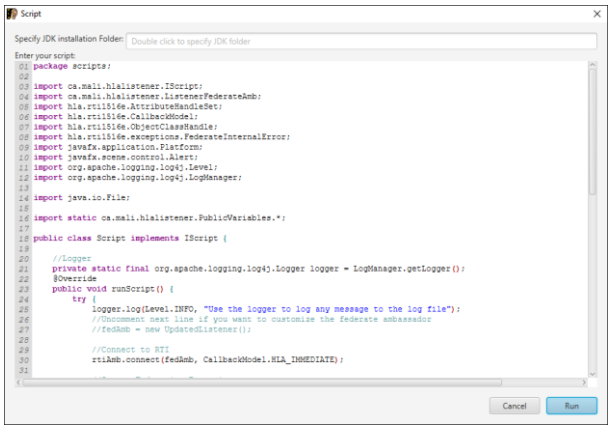

Figure 6 The script window can be used to interact with RTI through code.

5. HLA LISTENER SOURCE CODE

HLA Listener is published publicly under Apache License agreement. The source code and binaries can be obtained through its GitHub web page: https://github.com/EMostafaAli/HlaListener.

6. CONCLUSION

Distributed simulation is a powerful and flexible framework that enhances flexibility, reusability, and integration of large scale simulations. Nonetheless, it has a very steep learning curve and its standard is written in rigid technical style. This complexity turns away industrial world from utilizing this simulation framework. In this paper, we presented "HLA Listener", a tool that encapsulates all services provided in the HLA distributed simulation through a graphical user interface. HLA Listener was developed with two use cases in mind. The first usage is oriented to a beginner who wants to explore HLA capabilities without going into much details about programming. Through the interface, the user is able to invoke calls and receive callbacks from the RTI, it can also be used to run a simulation with other federates. In addition, HLA Listener can be used by an experienced simulator to test and debug existing federates as it has a console to write code along the interface. We plan to extend HLA Listener by adding the capability to record user interactions with the interface and automatically generate the corresponding code.

REFERENCES

AbouRizk, S.M., 2010. Role of Simulation in Construction Engineering and Management. Journal of Construction Engineering and Management 136, 1140–1153. doi:10.1061/(ASCE)CO.1943-7862.0000220

Ali, M., Fagiar, M., Mohamed, Y., AbouRizk, S.M., 2014. Beyond classic models—design and development of a comprehensive earthmoving simulator, in: 14th International Conference on Construction Applications of Virtual Reality in Construction and Conference on Islamic Architecture. Sharjah, UAE.

Boer, C.A., Bruin, A.D., Verbraeck, A., 2006a. Distributed Simulation in Industry - A Survey Part 1 - The Cots Vendors, in: Simulation Conference, 2006. WSC 06. Proceedings of the Winter. Presented at the Simulation Conference, 2006. WSC 06. Proceedings of the Winter, pp. 1053–1060. doi:10.1109/WSC.2006.323194

Boer, C.A., Bruin, A.D., Verbraeck, A., 2006b. Distributed Simulation in Industry - A Survey Part 2 - Experts on Distributed Simulation, in: Simulation Conference, 2006. WSC 06. Proceedings of the Winter. Presented at the Simulation Conference, 2006. WSC 06. Proceedings of the Winter, pp. 1061–1068. doi:10.1109/WSC.2006.323195

Boer, C.A., Bruin, A. de, Verbraeck, A., 2008. Distributed simulation in industry - a survey Part 3 - the HLA standard in industry, in: Simulation Conference, 2008. WSC 2008. Winter. Presented at the Simulation Conference, 2008. WSC 2008. Winter, pp. 1094–1102. doi:10.1109/WSC.2008.4736178

Borshchev, A., Karpov, Y., Kharitonov, V., 2002. Distributed simulation of hybrid systems with AnyLogic and HLA. Future Generation Computer Systems, Selected Papers presented at the 6th Int. Conf. on Parallel Computing Technologies (PaCT-2001) 18, 829–839. doi:10.1016/S0167-739X(02)00055-9

Bruzzone, A.G., Longo, F., 2013. 3D simulation as training tool in container terminals: The TRAINPORTS simulator. Journal of Manufacturing Systems 32, 85–98. doi:10.1016/j.jmsy.2012.07.016

Fujimoto, R.M., 2003. Distributed simulation systems, in: Simulation Conference, 2003. Proceedings of the 2003 Winter. Presented at the Simulation Conference, 2003. Proceedings of the 2003 Winter, p. 124–134 Vol.1. doi:10.1109/WSC.2003.1261415

Gerhardt-Powals, J., 1996. Cognitive engineering principles for enhancing human-computer performance. International Journal of Human-Computer Interaction 8, 189–211. doi:10.1080/10447319609526147

Goti, A., 2010. Discrete event simulations. Sciyo, Rijeka.

Ham, W.K., Kwon, Y., Park, S.C., 2014. Combat Simulation Framework Including Continuous Detection System. International Journal of Simulation Modelling 13, 395–408. doi:10.2507/IJSIMM13(4)1.262

Helbing, D., 2012. Agent-Based Modeling, in: Helbing, D. (Ed.), Social Self-Organization, Understanding Complex Systems. Springer Berlin Heidelberg, pp. 25–70.

Hibino, H., Yura, Y., Fukuda, Y., Mitsuyuki, K., Kaneda, K., 2002. Manufacturing Modeling Architectures: Manufacturing Adapter of Distributed Simulation Systems Using HLA, in: Proceedings of the 34th Conference on Winter Simulation: Exploring New Frontiers, WSC '02. Winter Simulation Conference, San Diego, California, pp. 1099–1107.

IEEE Standard for Modeling and Simulation (M\&S) High Level Architecture (HLA)-- Federate Interface Specification, 2010. . IEEE Std 1516.1-2010 (Revision of IEEE Std 1516.1-2000) 1,378. doi:10.1109/IEEESTD.2010.5557728

Jennings, N.R., 2000. On agent-based software engineering. Artificial Intelligence 117, 277–296. doi:10.1016/S0004-3702(99)00107-1

Lendermann, P., Heinicke, M.U., McGinnis, L.F., McLean, C., Strassburger, S., Taylor, S.J.E., 2007. Panel: distributed simulation in industry - a real-world necessity or ivory tower fancy?, in: Simulation Conference, 2007 Winter. Presented at the Simulation Conference, 2007 Winter, pp. 1053–1062. doi:10.1109/WSC.2007.4419704

Möller, B., Löfstrand, B., Karlsson, M., 2007. An overview of the HLA evolved modular FOMs, in: Simulation Interoperability Workshop, Spring.

Möller, B., Morse, K.L., Lightner, M., Little, R., Lutz, R., 2008. HLA evolved–a summary of major technical improvements, in: Proceedings of 2008 Spring Simulation Interoperability Workshop, 08F-SIW-064.

Page, R.L., 2000. Brief history of flight simulation. SimTecT 2000 Proceedings 11–17.

Schulze, T., Strassburger, S., Klein, U., 1999. Migration of HLA into Civil Domains: Solutions and Prototypes for Transportation Applications. SIMULATION 73, 296–303. doi:10.1177/003754979907300506

Staggers, N., Kobus, D., 2000. Comparing Response Time, Errors, and Satisfaction Between Text-based and Graphical User Interfaces During Nursing Order Tasks. Journal of the American Medical Informatics Association 7, 164–176. doi:10.1136/jamia.2000.0070164

Taylor, S.J.E., Bruzzone, A., Fujimoto, R., Gan, B.P., Strassburger, S., Paul, R.J., 2002. Distributed simulation and industry: potentials and pitfalls, in: Simulation Conference, 2002. Proceedings of the Winter. Presented at the Simulation Conference, 2002. Proceedings of the Winter, pp. 688–694 vol.1. doi:10.1109/WSC.2002.1172948

Turner, S.J., Cai, W., Gan, B.P., 2000. Adapting a supply-chain simulation for HLA, in: Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications, 2000. (DS-RT 2000). Proceedings. Presented at the Fourth IEEE International Workshop on Distributed Simulation and Real-Time Applications, 2000. (DS-RT 2000). Proceedings, pp. 71–78. doi:10.1109/DISRTA.2000.874065

Wooldridge, M., 1997. Agent-based software engineering. IEE Proceedings - Software Engineering 144, 26–37. doi:10.1049/ip-sen:19971026

Zacharewicz, G., Deschamps, J.-C., Francois, J., 2011. Distributed simulation platform to design advanced RFID based freight transportation systems. Computers in Industry, Special Issue: Grand Challenges for Discrete Event Logistics SystemsGrand Challenges for Discrete Event Logistics Systems 62, 597–612. doi:10.1016/j.compind.2011.04.009

AUTHORS BIOGRAPHY

Mostafa Ali is a PhD candidate at the Department of Civil and Environmental Engineering at the University of Alberta. His e-mail address is MostafaAli@ualberta.ca.

Yasser Mohamed is an Associate Professor in the Department of Civil and Environmental Engineering at the University of Alberta. His research interest includes modeling and simulation of construction processes and the integrating of simulation models and knowledge engineering tools into construction management and decision making processes. His e-mail address is yaly@ualberta.ca.