©2016 Published in 4th International Symposium on Innovative Technologies in Engineering and Science 3-5 November 2016 (ISITES2016 Alanya/Antalya - Turkey)

HIMES: HLA Interface for Mathematical and Engineering Simulation Tools

*[1]Bilge Kaan Gorur
[1]Roketsan A.S., PK 30, Elmadag, 06780, Ankara, Turkey

Abstract

Evolving technology and large scale simulations that require high computational power made distributed computing indispensable. Developing distributed simulations has numerous advantages, such as time efficiency that is gained by parallel development, and loosely coupled subsystems that can be reused in different projects. Therefore, some interoperability and reusability standards have been proposed in last decades. High level architecture (HLA) and distributed interactive simulation (DIS) are the most known interoperability standards. Although this kind of standards are well-known and widely used by especially software and computer science researchers, other disciplines do not use them so often, because popular engineering simulation tools do not provide an interface to those standards. In this paper, we present a distributed simulation interface that has been developed using a few open source software. The interface that we propose can connect model based simulation development tools with HLA standard.

Key words: Distributed simulation, interoperability, reusability, model-based design, high level architecture

1. Introduction and Background

Parallel and distributed simulation (PADS) is one of the most studied topics in modeling and simulation (MODSIM) world. There are two main factors that make researchers to need PADS. The first one is insufficiency of the computational power of single computing nodes. Therefore, researchers need many computational resources when they would like to simulate their large scale models, such as computational fluid dynamics. To overcome this problem, researchers bring computing nodes together via a network and leverage the total computational power of them. The second factor is the necessity of interoperable simulations. Researchers needed to run their models together with other models that have been developed by various researchers, departments, companies and countries to extend their work [1]. These two factors made PADS more popular in last two decades [2]. This paper focuses the interoperability and reusability of distributed simulations rather than parallel simulation techniques.

To handle interoperability and reusability challenges, MODSIM researchers created some standards, architectures and protocols, such as high level architecture (HLA), data distribution service (DDS) and distributed interactive simulation (DIS) [3, 4, 5]. Simulation developers who can adapt their model into these standards are able to run different models together. Interoperability standards basically define the communication and interaction rules among distributed simulation nodes that are known as federates in HLA. In this study, we have used HLA for our distributed simulation infrastructure, because we think that HLA is very popular and

*Corresponding author: Address: Roketsan Inc., PK 30, Elmadağ, 06780, Ankara TURKEY E-mail address: kaan.gorur@roketsan.com.tr, Phone: +903128605500

widely used by the community. A brief comparison between HLA and similar approaches is given in Table 1 and 2. The detailed comparisons can be also found in [6, 7, 8].

Table 1. Main differences between HLA and DIS [6]

HLA	DIS
A central manager, namely RTI, exists	No central server
Simulation application is independent from communication protocol.	Transmission of information packets are performed with a specified protocol, namely protocol data units (PDU)
More scalable thanks to data declaration management	Less scalable because of the dense network traffic
Wider range of simulations (human-in-the-loop, constructive, real-time, non-real time)	Focuses on real-time virtual simulations
Reliable TCP/IP communication	Unreliable UDP communication
Point to point communication	Broadcasting for publication

Table 2. Main differences between HLA and DDS [7, 8]

HLA	DDS
APIs for federation save/restore and synchronization point	No APIs for federation save/restore and synchronization point
API for time management	No API for time management
Less QoS (quality of service) policies	More QoS policies
Dependency on federation object model (FOM)	No dependency of global knowledge
Region management mechanism	Content-based subscription
Static declaration of FOMs	Fully dynamic discovery

The first version of HLA standard, as known as HLA 1.3, was sponsored by US Defense Modeling and Simulation Office and published in 1998 [3]. Then, HLA was superseded by IEEE and published as IEEE 1516 standard in 2000. This HLA version was extended to the current version of HLA in 2010 (also known as HLA 1516-2010 or HLA Evolved) [4]. The detailed history and evolution of distributed simulation standards can be found in [9]. HLA standard proposes to have a Run-Time Infrastructure (RTI) that manages the coordination of federates in a distributed simulation system. For interoperable simulations, federates have to make an agreement that is called federation agreement on how they are going to share objects and interactions. The federation agreement includes federation object model that defines objects, interactions and data types [10].

HLA's functional interfaces between RTI and federates are arranged into these seven groups [4]:

- **Federation management** is responsible for federation lifecycles. It provides services for creating, modifying, synchronizing and deleting a federation that federates can join.
- **Declaration management** is responsible for regulating publishing and subscribing actions. It provides services for declaring what kind of objects or interactions will be produced and

consumed by federates.

- **Object management** is interested in registering, discovering and deleting object instances; updating and reflecting objects' attributes.
- **Ownership management** is responsible for managing the responsibility of updating and deleting object instances.
- **Time management** provides services for advancing simulation time of federates. Conservative or optimistic time management techniques can be preferred by federates.
- **Data distribution management** is interested in efficient routing of data by organization of regions so that federates can survive irrelevant data.
- **Support services** are interested in miscellaneous services, such as starting-up/shutting-down RTI, manipulating regions and making callbacks from RTI to federates.

The HLA standard provides application programmer's interfaces (API) for some well-known object oriented programming languages, including Java and C++, so object-oriented software developers can develop HLA compliant models. However, developers who should work with other kind of languages/tools cannot connect their models to an HLA infrastructure as easy as object-oriented developers. For instance, employing HLA in advanced engineering tools, such as Matlab and Scilab, requires an extra interface to HLA. Similarly, Simulink and Xcos that are model-based development tools of Matlab and Scilab require that kind of interface, too. These two tools allow developers to implement simulations with model-based design techniques by using block diagrams, because model-based design makes developer to create and modify models faster [11, 12]. Therefore, creating an interface to HLA from engineering software and their model-based development tools is crucial for interoperability of engineering simulations.

In this paper, we present a distributed simulation interface that can connect some well-known engineering tools and HLA standard. This interface, namely HLA Interface for Mathematical and Engineering Simulation Tools (HIMES), has been developed by using open source software. Thanks to HIMES, developers can connect and run their models that have been developed in Matlab, Simulink, Scilab and Xcos together via HLA standard. The rest of this paper is structured as follows: in Section 2 some of the related works in the literature and example usage of HLA are pointed out. Next section gives detailed information about how we have implemented HIMES. Finally, we discuss the advantages of HIMES and our future plans in Section 4.

2. Related Work

Interoperability and reusability have been widely studied by MODSIM community for several years. Although numerous studies can be found in the literature, we gave a place to some public and large scale examples here. One of the large scale distributed simulation studies in NATO can be found in [13], also known as Exercise First Wave. 15 sites from 7 countries (US, Canada, Netherlands, Italy, France, Germany and UK) participated in this exercise that was performed for aircrew mission training. Since Exercise First Wave was one of the earliest and largest exercises with HLA, it enabled testing of HLA in a unique and demanding environment [13].

Regarding to HIMES-like tools, ForwardSim's HLA Toolbox (for Matlab) and HLA Blockset (for Simulink) makes a connection between RTI and Matlab/Simulink [14]. Both of them were developed to connect interoperable and reusable simulation models with advanced engineering tools. A similar study for Scilab can be found in [15]. In that study, Theppaya et al. proposed to integrate RTI services with Scilab. Ravn et al. also provides a DIS interface to Simulink for simulation visualization [16]. Differently from those studies, HIMES provides an interface to Scilab's model based development tool Xcos that is a widely used open source alternative to Simulink. HIMES serves some Xcos blocks for developers to use HLA services in their simulation models. In short, HIMES can be used with any of Matlab, Scilab, Simulink and Xcos; and can be extended for any other engineering tools that can make Java function calls.

Parallel and distributed computation studies for Scilab and Xcos are in very early steps. The paper in [17] is one of the first distributed simulation studies for Scilab. Mukbil et al. implemented a networking module that allows communication between Scilab models via UDP messages. To this end, we think that our study is also going to pave the way for parallel and distributed simulation studies for Scilab and Xcos.

3. HLA Interface for Mathematical and Engineering Simulation Tools (HIMES)

As we have mentioned in previous sections, a bridge between engineering simulation tools and distributed simulation standards is necessary for interoperability and reusability. So, we have being developed an interface for Matlab, Scilab and their model based design tools Simulink and Xcos. Since we aimed to develop a flexible HLA interface, HIMES can be also adapted for other tools easily. In case developers would like to use HIMES with other tools, they should do only these two things: importing HIMES library (a jar file) into their environment and creating the code or block that calls the related HIMES function which is a bridge to Portico RTI from the tool that he/she would like to use.

3.1. Architectural View of HIMES

One of the main focuses of this work is showing that this kind of interface can be done by using open source or free software. To this end, we preferred to use an open source RTI library, Portico RTI [18]. We have implemented HIMES with Java programming language because of three main reasons. The first one is Portico RTI has been developed with Java and C++, so we were free to choose one of them. Secondly, Java is a platform-independent language and can be used in Windows or Unix-based computers. The last reason is that the tools like Matlab, Scilab and their simulation modules (Simulink and Xcos) can import libraries and call functions that have been developed with Java. Therefore, integrating HIMES with both Matlab and Scilab can be done in a few steps easily. Fig. 1 shows how the HIMES takes a place in the distributed simulation environment.

In our design, a federate in the system (master federate in Fig. 1) should be different from the others and manages the simulation. For simplicity reason we provide a basic master federate that creates and destroys the federation in Java. After it starts to run, the other federates are waited to be joined the federation. HIMES has also interface for federation management service and allows developers to implement their own master federate.

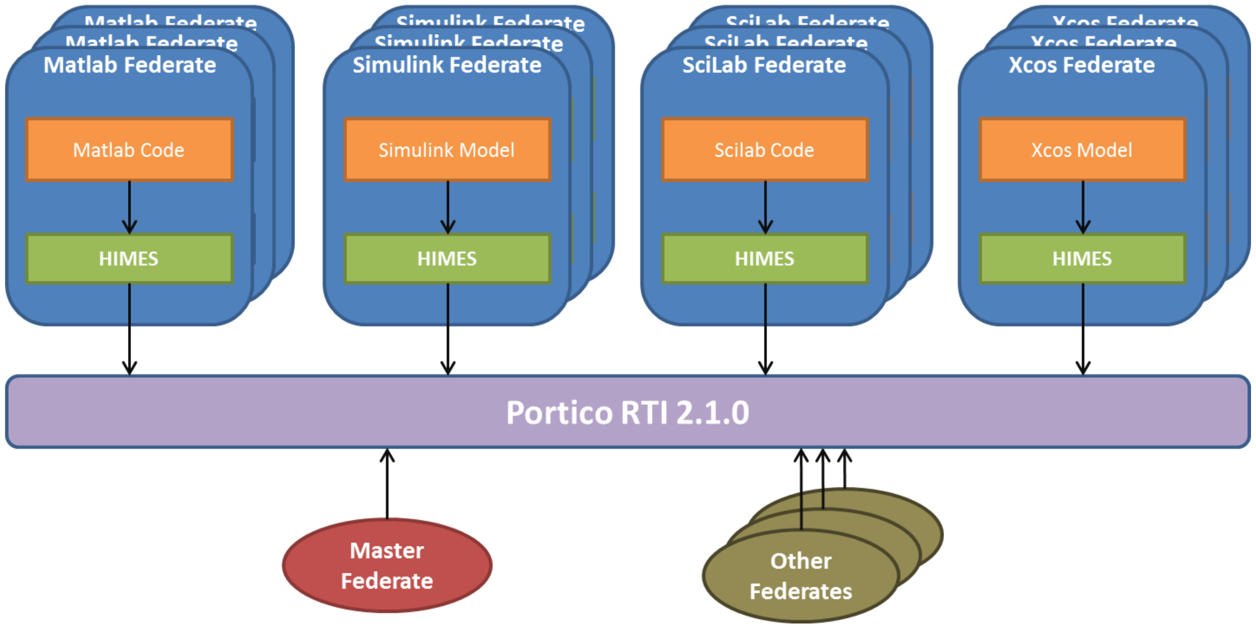

Figure 1. Architectural view of distributed simulation components and HIMES

3.2. HLA Services Provided by HIMES

In HIMES, we did not allow developers to use all of the HLA services, because some of them will not be needed so often by them. The allowed HLA functions can be seen in Table 3. In this table, function type refers to the caller of the function. The function type that is specified as “Call” means that it is called by federates and responded by RTI. “Callback” type is the opposite, that is, the function is called by RTI to the federate ambassador.

Federates interact with each other according to the FOMs that should be defined before the federation execution starts. The FOM should be specified in a text file that is prepared according to the format that Portico RTI recognizes. This file should be placed in the same directory with models that have been developed by Matlab, Scilab, Simulink or Xcos.

HIMES provides an interface for model based development tools of Matlab and Scilab, too. To this end, we implemented some HLA blocks for Simulink and Xcos. An example usage of HIMES with Scilab and Xcos is given in Fig. 2 and 3. Fig. 2 is the Scilab code that manages the initialization and termination of an Xcos federate. The federate firstly joins the federation that has been already started by the master federate. The declaration of publishing and subscribing attributes are also specified in this code. In this example, the federate publishes and subscribes to the same attribute. Next, an object is registered and the RTI is notified about it. After these operations, the simulation that has been modeled with Xcos as in Fig. 3 is executed. The green colored blocks are HLA related blocks. In this dummy model, the “hlaUpdateAttributeValues” block updates objectHandle’s attributeHandle to the output of the square root of a sinus generator block at every time step. Next, advancing federate’s time is requested from RTI by “hlaAdvanceTime” block. When the RTI lets the federate advance time, the federate is free to process the next time step. During the execution of the “hlaAdvanceTime” block, callbacks from RTI to the federate ambassador are also performed. Lastly, the execution of Xcos model is terminated and the federate resigns federation as in Fig. 2.

3.3. Limitations and Missing Services

Although we have tried to design HIMES as a flexible tool, it comes with some limitations that we have not handled, yet. One of them is the lack of ownership management service of HLA. The only reason of this lack is that we do not transfer the ownership of objects and attributes in our experiments. Similarly, we have not implemented some other services of HLA that are unnecessary for us, yet. Optimistic time management and interaction management are some of these unimplemented services.

The other limitation of HIMES is forcing developers to run their models with the same ordinary differential equation (ODE) solver. Since ODE solvers are used in time management of the simulations, we had to force them to be the same for synchronizing global time of the simulation in a proper way.

4. Conclusion and Future Work

In this study, we have developed an interface to some well-known advanced engineering tools for being able to create distributed simulations. We preferred to employ HLA standard to make our simulations interoperable and reusable. One of the most prominent features of HIMES is that it has been implemented using open-source software, such as Portico RTI and Scilab. Besides, HIMES is integrated with model-based design tools Simulink and Xcos that enable rapid simulation development. Therefore, developers who prefer to use those tools are able to integrate their models with HLA and interoperate various models. Similar to Matlab and Scilab, other engineering tools can be easily integrated with HIMES, if they can call Java functions.

Currently HIMES is being used as a prototype and we are planning to complete integration and system tests of HIMES in the future. Next, we are going to implement some interfaces for advanced time management mechanisms to run simulations faster. For instance, Time Warp algorithm [19] is a scalable time management mechanism and simulations that enable it can run faster than usual simulations in some cases. On the other hand, providing interoperability of models that have different ODE solvers is also a challenge that we are planning to overcome in the next versions of HIMES.

In addition to technical improvements, we are going to handle distributed simulation development processes, such as HLA FEDEP (Federation Development and Execution Process) [20] and DSEEP (Distributed Simulation Execution and Engineering Process) [21]. Our goal is making HIMES to allow developing distributed simulations conforming to these processes.

Table 3. Interfaced HLA services in HIMES

HLA Service Name	Function Type	FunctionName
Federation Management	Call *	createFederationExecution
	Call	joinFederationExecution
	Call	resignFederationExecution
	Call	destroyFederationExecution
	Call	registerFederationSynchronizationPoint
	Callback **	synchronizationPointRegistrationSucceeded
	Callback	announceSynchronizationPoint
	Call	synhronizationPointAchieved
	Callback	federationSynchronized
Time Management	Call	timeAdvanceRequest
	Callback	timeAdvanceGrant
	Call	nextEventRequest
	Call	queryFederateTime
	Call	queryLookahead
	Call	modifyLookahead
	Call	queryLBTS
	Call	queryMinNextTimeEvent
	Call	enableTimeConstrained
	Call	enableTimeRegulation
Declaration Management	Call	getObjectClassHandle
	Call	getAttributeHandle
	Call	publishObjectClass
	Call	subscribeObjectClassAttributes
	Callback	startRegistrationForObjectClass
	Callback	stopRegistrationForObjectClass
	Call	unpublishObjectClass
	Call	unsubscribeObjectClass
	Call	getInteractionClassHandle
Object Management	Call	registerObjectInstance
	Call	discoverObjectInstance
	Callback	turnUpdatesOnForObjectInstance
	Call	deleteObjectInstance
	Call	requestClassAttributeValueUpdate
	Callback	provideAttributeValueUpdate
	Call	requestObjectAttributeValueUpdate
	Call	updateAttributeValues
	Callback	reflectAttributeValues
Data Distribution Management	Call	createRegion
	Call	notifyAboutRegionModification
	Call	deleteRegion
Support Services	Call	evokeCallbacks
	Call	evokeMultipleCallbacks

* *Federate's function. This function is called by any federates*
** *Callback function. This function is called by RTI to notify federates*

```
//  Import HLA library and HLA interface for Scilab
javaclasspath(strcat([pwd(), "\lib\portico.jar"]));
javaclasspath(strcat([pwd(), "\lib\scilabHLAInterface.jar"]));
jimport rs.hla.ScilabFederate;
jimport rs.hla.ScilabFederateAmbassador;

//  Create a new federate object
federate = ScilabFederate.new();
fomsPath = {'foms/ObjectModel.xml'};

//  Create object and attribute handles for publish/subscribe mechanisms.
objectClassHandle = jinvoke(federate, 'getObjectClassHandle',
                                                'HLAobjectRoot.MyObject');
attributeHandle = jinvoke(federate, 'getAttributeHandle',
objectClassHandle, 'myAttribute');

//  Join federation; declare publishing and subscribing objects
jinvoke(federate, 'joinFederation', 'FEDERATION_NAME', 'ScilabFederate',
                                                            fomsPath);
jinvoke(federate, 'publish', objectClassHandle, attributeHandle);
jinvoke(federate, 'subscribe', objectClassHandle, attributeHandle);

//  Notify RTI about an object has been created.
objectInstanceHandle = jinvoke(federate, 'registerObjectInstance',
objectClassHandle, attributeHandle, updatedValue);

//  Run HLA compliant Xcos model
importXcosDiagram("Model.xcos");
scicos_simulate(scs_m);

//  Resign federation
jinvoke(federate, 'resignFederation', 'FEDERATION_NAME');
```

Figure 2. The Scilab code that initializes an Xcos federate

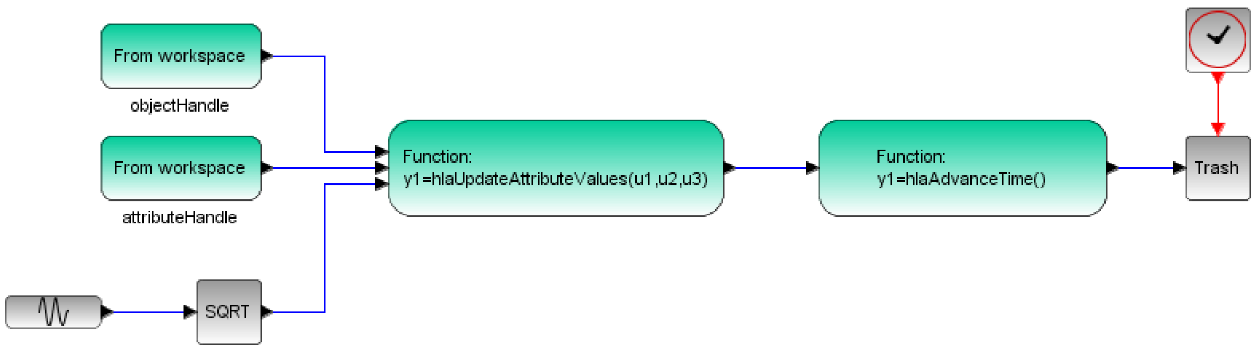

Figure 3. A sample block diagram of HLA compliant Xcos model

References

[1] Fujimoto RM. Parallel and distributed simulation systems. 1st ed. New York: John Wiley & Sons, Inc; 1999.
[2] Fujimoto RM. Research challenges in Parallel and Distributed Simulation. ACM Transactions on Modeling and Computer Simulation; 2016.
[3] Department of Defense, Defense Modeling and Simulation Office (DMSO). High Level Architecture Interface Specification, Version 1.3; 1998.
[4] IEEE 1516-2010. IEEE Standard for Modelling and Simulation (M&S): High Level Architecture (HLA); 2010.
[5] IEEE 1278. IEEE Standard for Distributed Interactive Simulation (DIS); 1995.
[6] Jense JG, Kuipers NHL, Dumaij ACM. DIS and HLA: Connecting people, simulations and simulators in the space, military and civil community. In Proc. 48th International Astronautical Congress; 1997
[7] Corsaro A, Martines-Salio JR. Distributed simulations with DDS and HLA. PrismTech Webcast. http://www.prismtech.com/. As of 11th Oct 2016.
[8] Joshi R, Castellote GP. A comparison and mapping of data distribution service and high-level architecture. Real-Time Innovations Inc.; 2006.
[9] Hollenbach JW. Inconsistency, Neglect, and Confusion; A Historical Review of DoD Distributed Simulation Architecture Policies. In Proc. Spring Simulation Interoperability Workshop, San Diego, CA; 2009.
[10] Department of Defense, Defense Modeling and Simulation Office (DMSO). RTI 1.3 – Next Generation Programmer's Guide Version 3.2; 2000.
[11] Hutchinson J, Whittle J, Rouncefield M. Model-driven engineering practices in industry: Social, organizational and managerial factors that lead to success or failure. Science of Computer Programming; 2014.
[12] Schmidt DC. Model-driven engineering. IEEE Computer; 2006; 39:2.
[13] Greschke DA, Cerutti S. Aircrew Mission Training via Distributed Simulation (MTDS) Development of the Multi-Country Complex Synthetic Environment. NATO RTO HFM Symposium on Advanced Technologies for Military Training; 2003.
[14] ForwardSim. http://www.forwardsim.com/. As of 19th Sep 2016.
[15] Theppaya T, Tandayya P, Jantaraprim C. Integrating the HLA RTI Services with Scilab. Sixth IEEE International Symposium on Cluster Computing and the Grid; 2006.
[16] Ravn O, Andersen NA. Using DIS for Linking Simulation and Animation in Simulink and VRML. In Proc. 8th IFAC Symposium on Computer Aided Control System Design CACSD 2000; 2000.
[17] Mukbil A, Stroganov P, Durak U, Hartmann S. Towards a Distributed Simulation Toolbox for Scilab. Workshop der ASIM/GI Fachgruppen STS und GMMS; 2016.
[18] The Portico Project. http://www.porticoproject.org. As of 19th Sep 2016.
[19] Jefferson DR. Virtual Time. ACM Transactions on Programming Languages and Systems (TOPLAS); 1985.
[20] IEEE 1516.3-2003. IEEE Recommended Practice for High Level Arcihtecture (HLA) Federation Development and Execution Process (FEDEP); 2003.
[21] IEEE 1730-2010. IEEE Recommended Practice for Distributed Simulation Execution and Engineering Process (DSEEP); 2010.