Integrating the HLA RTI Services with Scilab

Thitima Theppaya
Pichaya Tandayya
Chatchai Jantaraprim
Department of Computer Engineering
Faculty of Engineering
Prince of Songkla University
P.O. Box 2, Kohong, Hat Yai, Songkhla 90112 Thailand
s4612026@psu.ac.th, pichaya@coe.psu.ac.th, cj@coe.psu.ac.th

Abstract

This paper describes the integration of the High Level Architecture (HLA), an IEEE standard for distributed interactive simulation, with a scientific software package (Scilab) and its use for collaborative simulation development. This work is an on-going work that aims to facilitate the interoperability and reusability for Scilab simulation models. Integrating HLA with the engineering and scientific software package will enable users to apply simulation techniques to larger and more complex interactive and independent simulation models using networked computers, including virtual environments. As the HLA employs the technique of callback functions as a means for communication amongst the Run-time Infrastructure (RTI), the HLA middleware, and the simulation nodes, the integration was not simple. The paper also gives an example application showing how the HLA services are used to enable Scilab to construct distributed interactive simulations.

1. Introduction

This paper presents an on-going work that supports the collaborative simulation development in the field of distributed simulation using the High Level Architecture (HLA) [1-3], an IEEE standard, to facilitate interoperability and reusability of Scilab [4] simulation models.

The development of complex discrete event simulation applications usually requires collaborative effort from possibly geographically located simulation systems and researchers with multidisciplinary knowledge and expertise. The HLA provides an approach for constructing large-scale distributed simulations. Work on this area can address challenges in developing large-scale distributed simulations on the Grid.

The implementation of distributed interactive simulation systems using HLA requires a high level of programming skills from the developers. Unfortunately, most simulation developers are not computer engineers or computer scientists. Therefore, they do not have deep programming knowledge and usually prefer to use mathematical and engineering tools including Scilab and Matlab [3] that provide ready-made functions.

Integrating HLA with such engineering and scientific software packages will enable users to apply simulation techniques to larger and more complex interactive simulation models using networked computers, including virtual environments. Some work has been done on integrating Matlab with HLA [2,3]. However, Matlab is a commercial application. We chose Scilab in this work because it is an open source and it has not yet been enhanced for distributed interactive simulation.

Developed by the National Institute for Research in Computer Science and Control (INRIA) and Ecole National des Ponts et Chaussees (ENPC) in France, and now maintained and developed by Scilab Consortium, Scilab is a scientific software package for analyzing and designing solutions to mathematical and engineering problems [4]. Scilab provides a powerful open computing environment for engineering and scientific applications. As it is distributed freely as an open source via the Internet, Scilab is popularly being used in educational and industrial environments around the world.

Scilab is suitable for creating from basic to complex simulation models utilizing hundreds of mathematical and engineering functions, with the developer requiring only a limited knowledge of programming [4,5]. Scilab is also available on multiple platforms, and is easily integrated with various external language programs

Proceedings of the Sixth IEEE International Symposium on Cluster Computing and the Grid Workshops (CCGRIDW'06)
0-7695-2585-7/06 $20.00 © 2006 IEEE

that are possible to be added interactively. In addition, the user can define new data types and operations on the data types of the Scilab open system by using overloading. Although Scilab only has support for parallel computing (Parallel Scilab using PVM) [4], it is possible to enhance Scilab to be used for distributed simulation. Scilab has considerably suitable supporting tools for developing distributed interactive simulation including 2-D and 3-D graphics, and animation.

The prominent services of the HLA include Federation Management, Data Declaration, Data Distribution, Object Management, Ownership Management and Time Management [1,6]. They will provide support for collaborative simulation development in Scilab while facilitating interoperability and reusability of simulation models. These services are not provided under the parallel computing framework of MPI [7] or PVM that aims for solving a single unified problem. The advantages of using the HLA are dealt with the ability of linking independent complex components, object-orientation, reusability, object ownership transfer, and simulation time environment.

2. High Level Architecture (HLA)

Developed by the Defense Modeling and Simulation Office (DMSO), the HLA consists of [6]:

- Basic simulation rules defining a federate, a federation, and their interaction through the Run-time Infrastructure (RTI) [1,6].
- A data format specification, the Object Model Template (OMT) [8].
- An interface specification [6].

An HLA simulation node is called a federate. A collection of federates that builds a networked simulation system is called a federation. The federates interact with one another through the RTI. A federate consists of a simulation application and an RTI library (*libRTI*) for interfacing with the RTI [1,9]. The structure of an HLA federation is shown in Figure 1.

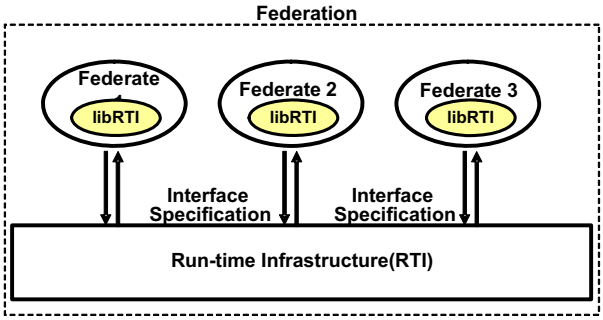


Figure 1. The structure of a HLA federation [1].

The HLA RTI uses callback functions as a means for the communication of distributed simulation. The RTI library provides the federate a *FederateAmbassador* and an *RTIambassador* as shown in Figure 2. By this way, the federate can remotely activate a function located on the RTI side through the *RTIambassador* and the RTI then remotely callback to a function of the federate through the *FederateAmbassador*.

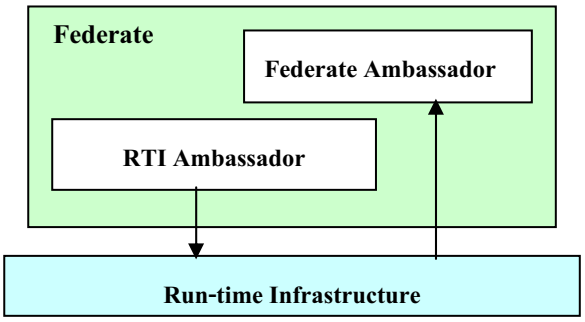


Figure 2. The interaction between a federate and the RTI [1].

3. Scilab Overview

Scilab consists of three parts: an interpreter, libraries of functions (Scilab procedures), and libraries of Fortran and C routines [10]. Scilab has an open programming environment where the user can arbitrarily create functions and libraries of functions. Functions are recognized as data objects in Scilab. Therefore, functions in Scilab can be manipulated or created as other data objects. This means that Scilab functions can be passed as input or output arguments of other functions. In addition Scilab allows on-line creation of functions as it supports a character string data type. Finally, Scilab can be interfaced with Fortran or C subprograms. This allows use of external standardized packages and libraries in the interpreted environment of Scilab. Figure 3 shows the structure of Scilab functions and how to add more functions.

Scilab supports several techniques for interfacing with external programs. The techniques include *dynamic links*, *intersci,* and *mexfiles* [10]. The *intersci* tool was used in this work because it has support functions for interfacing to C or Fortran. In this way, it reduces costs in interfacing and loading the external program into Scilab. As shown in Figure 4, a gateway file is to be created for transferring input and output between the external program and Scilab. *Intersci* creates a dynamic library for loading functions from the external program into Scilab. The user then calls the external functions via Scilab scripts or functions.

4. Integrating the HLA with Scilab

This section describes how we built our HLA module for Scilab. The Scilab used in this work is version 2.7 running on Linux Redhat 7.2. The DMSO RTI 1.3 version 6 for Linux was used in this work because it was the last freely available version before RTI was commercialized.

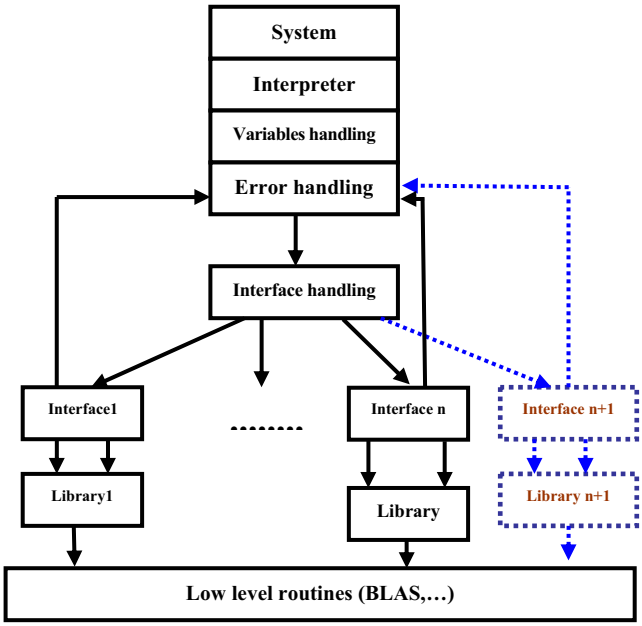


Figure 3. The structure of Scilab functions.

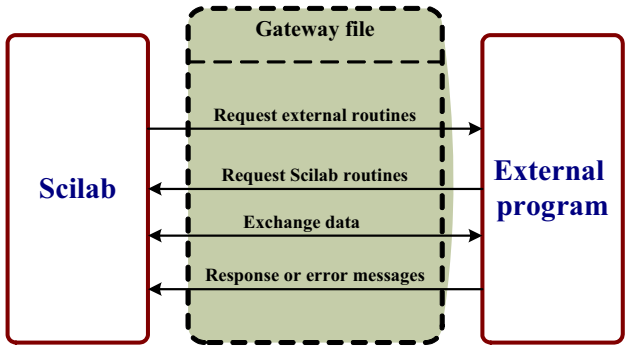


Figure 4. The interaction between Scilab and an external program via a gateway file [11].

4.1. Conceptual Design and Integration

A user may call Scilab modules by using the Scilab interfacing window or writing a script file. Each module in Scilab is independent. However, they are inherited from the same Object Root. The HLA module is added to Scilab as one of the Scilab modules by inheriting from the Object Root. A Scilab application can then call the HLA module for connecting to the RTI.

The HLA module in Scilab calls the *libRTI* in order to connect to the RTI. Figure 5 and Figure 6 show that the RTI interface in Scilab composing from two program files: the gateway file for calling the Call RTI part in the operation file that consequently calls the RTI via *RTIambassador*.

The interface in the operation file also includes the Callback Scilab part for the RTI to call the Scilab functions via *FederateAmbassador*. Function *Scirun* in Scilab was first used to work as a medium in passing the data from the external program to be shown on Scilab. However, it was found that although Function *Scirun* passes return data for the callback function, it quits itself and does not continue the rest instruction in the simulation script. Therefore, a special function has been created to replace *Scirun* for passing the return data for the callback function. The simulation then needs to periodically check for the return data.

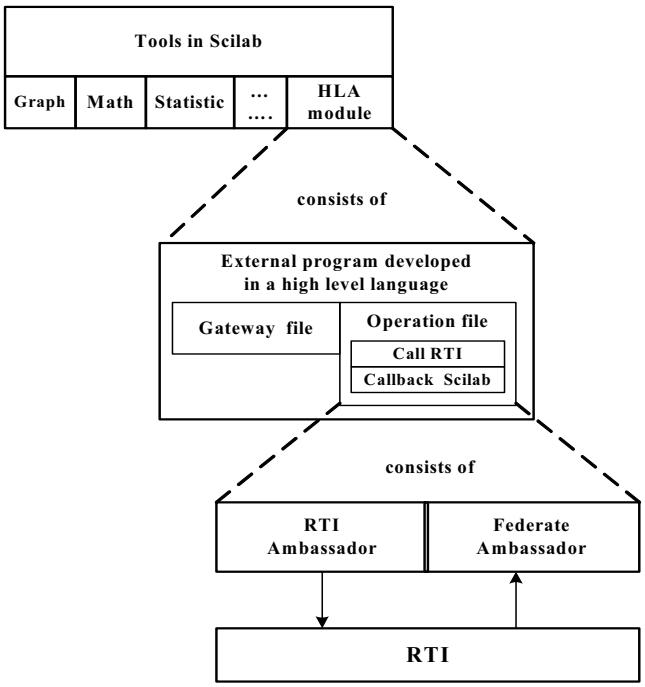


Figure 5. The RTI and Scilab interface.

4.2. HLA Services on Scilab

In this work, necessary HLA services have been developed, namely, Federation Management, Declaration Management, Object Management and Time Management. The added HLA services are adequate for running distributed simulation on Scilab.

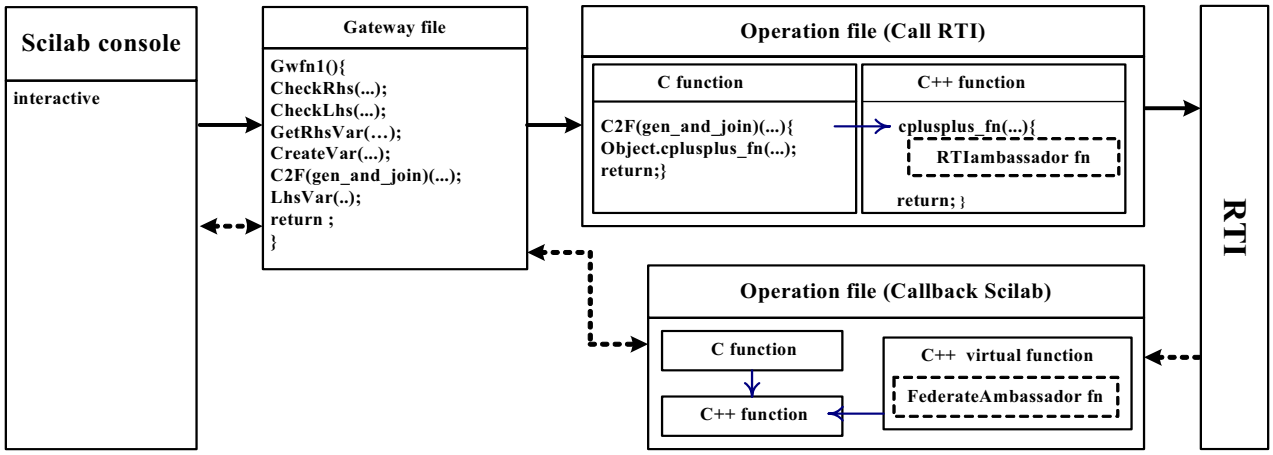


Figure 6. The detail of the RTI and Scilab interface.

This subsection describes the HLA services and their intgrated functions on Scilab.

4.2.1. Federation Management

The HLA RTI supports the Federation Management by ennabling the arbitrarily participation of the federates in the federation. A federate can join and leave at any time. In this work, Function *gen_and_join(FederateName, FederationName, fedfile)* is to be called by a Scilab federate from the Scilab interactive window for joining a federate. Likewise, similar functions for leaving the federation and destroying a federation are also provided.

4.2.2. Declaration Management and Object Management

The simulation in the HLA is object-oriented. The HLA RTI facilitates communication in the distributed simulation by using the Public-and-Subscribe mechanism. A federate can publish objects of classes that have been declared to the federation. Similarly, a federate can subscribe update attributes of any object of the declared classs. In this work, Functions *publish_object(ObjectTypeID,NumberOfAttribute)* and *subscribe_object(ObjectTypeID,NumberOfAttribute)* were created to enable the mechanism for Scilab federates.

Likewise, Functions *regist_object(ObjectTypeID)* and *update_attr(ObjectHandleID,Attr1,Attr2,…)* support the object registration, and attribute update. Through callback functions, the discovery of the object instant and the reflection of the object attributes will then automatically activated through the *FederateAmbassador* by the RTI. The simulation on Scilab will the use the supported functions such as Function *get_attr()* for showing the return data. Figure 7 shows the sequence diagram for the above services in the Object Management. When objects are no longer needed for the simulation, they can be deleted by the source federate and will then be removed from the subscribed federates.

Apart from objects, the other type of information passed in the federation is Interaction. The HLA Interaction is an event message with or without a parameter. An Interaction is different to an object that it has neither instance nor stored data once the message has been sent out. Similar to the management of objects, the Interaction management also uses the Public-and-Subscribe mechanism. Functions concerning sending and receiving the Interaction are also provided, i.e., Function *send_inter(InteractionHandleID,Value)* was used to send interaction update.

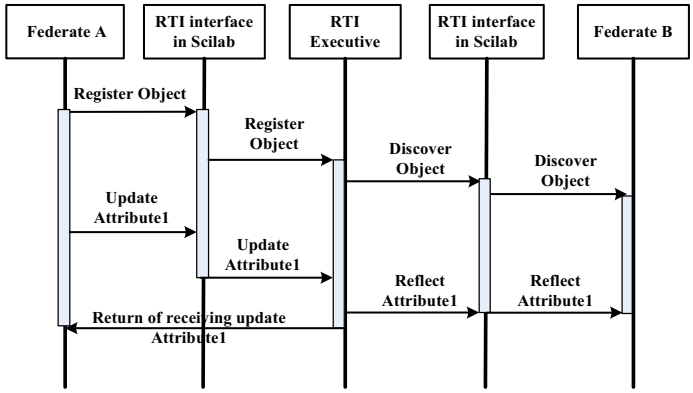


Figure 7. The sequence diagram for the Object Management.

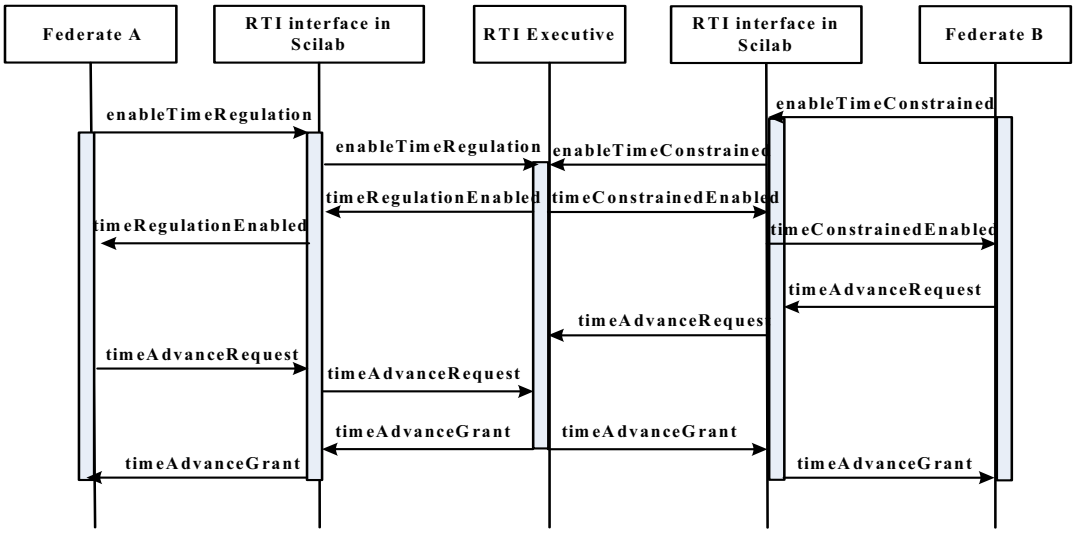


Figure 8. The sequence diagram for the Time Management in case of the time-stepped method.

4.2.3. Time Management

The Time Management includes the services that synchronises federates in the HLA federation. The RTI controls the time advancement of each federate. A federate that creates an event or sends a message with a time stamp to other federates is called a regulating federate. A constrained is a federate that receives events or messages with time stamps from other federates. The time advancement of a constrained federate is tied to that of regulating federates. In the time-stepped simulation, the Time Management process consists of two following phases.

(a) Time advancement request: a federate that wants to advance its local time calls the service Time Advance Request (TAR) [9] via the *RTIambassador* through the RTI.

(b) Time advancement grant: after receiving a time advance request from a regulating federate, the RTI calculates the Lower Bound on the Time Stamp (LBTS) [9] and then allows the time advancement of the requesting federate based on the LBTS. When a federate is allowed to advance its time, the RTI will call the service Time Advance Grant (TAG) [9] at the federate through the *FederateAmbassador*.

Figure 8 shows the sequence diagram of the Time Management in this work. Federate A is a regulating federate and Federate B is a constrained federate. After Federate B asks for advancing its time, the RTI will not allow the time advancement until Federate A asks for advancing its time that equals to or is more than the request of Federate B.

The functions on Scilab that enable the Time Management are Functions *en_time_reg(LookaheadValue)* and *en_time_const()* for assigning regulating and constrained federates, and Functions *time_step(TimestepValue)* and *time_adv_req()* for setting the step of time and requesting time advancement. The RTI uses callback functions for automatically granting time advancement. Similar functions are also provided in case of the event-driven simulation scheme, for example, *next_event_req (TimeValue)*.

5. Example Application: Ice Cream Federation

The interaction amongst federates in the HLA is made by federates share the attributes of objects and interactions in the federation. A testing federation was chosen to show how Scilab is used to construct distributed interactive simulation using HLA.

This testing federation is a simple ice cream distribution simulation. The federation contains two federates: *Ice Cream Distributor* and *Ice Cream Production*. The data distributed around consists of an object of Class *Ice cream* and an Interaction of Class *Message*. Table 1 shows the relationship amongst federates, object, interaction and time. The federation was designed to test the four implemented services of the HLA interfaced with a Scilab application.

Federate *Ice Cream Production* publishes Object *Ice Cream* created for supplying Federate *Ice Cream Distributor* and subscribes Interaction *Message* published by Federate *Ice Cream Distributor* for telling the demand for ice cream.

At the beginning of the simulation, the number of ice cream was initialized at 140. The duration for the simulation is 200 time units. Every 10 time units,

Federate *Ice Cream Production* produces 20 ice cream units and sends the current number of available ice cream to Federate *Ice Cream Distributor*. Federate *Ice Cream Distributor* consumes the ice cream and requests for more ice cream once the number is reduced to fewer than 20. The ice cream consumption is made by using a Scilab random function every 10 time units. Federate *Ice Cream Distributor* sends a message to Federate *Ice Cream Production* for telling the demand for ice cream. As a consequence, the number of ice cream in the Federate *Ice Cream Production* will be reduced.

Table 1. Relationship amongst the federates, object, interaction and time in the testing federation.

Object/Interaction/ Time / Federate	Object Ice cream	Interaction Message	Time Management
Ice cream Production	Publish	Subscribe	Time regulating
Ice cream Distributor	Subscribe	Publish	Time constrained

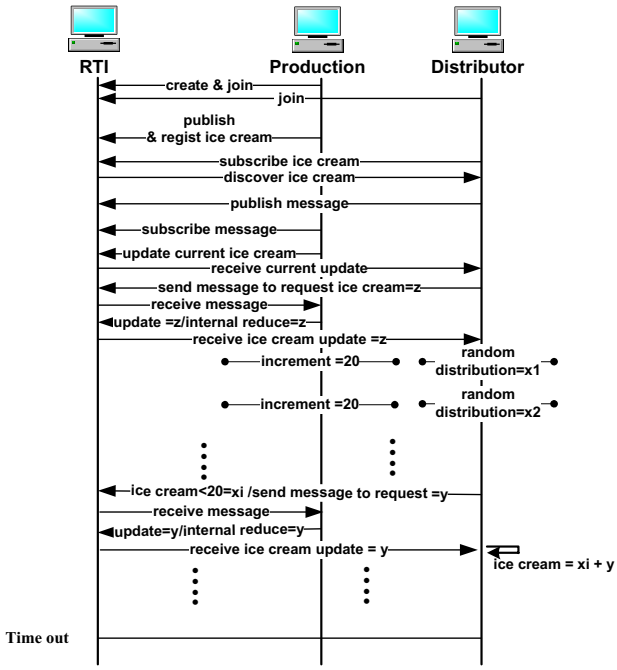


Figure 9. Sequence diagram of the ice cream simulation.

In case of time management, Federate *Ice Cream Production* is time regulating while Federate *Ice Cream Distributor* is time constrained. The time advancement of Federate *Ice Cream Production* is delayed by intention. Therefore, Federate *Ice Cream Distributor* must wait until the time advancement of Federate *Ice Cream Production* has been done before it can advance its own time. Figure 9 shows an example of a sequence diagram of the federation. Figure 10 shows the user interface of the testing simulation including Scilab console views and simulation reports. Users can trace the dialog display to check events occurred during the simulation or changes to the resources. Figure 11 shows the result of the simulation.

Using the testing scenario, it shows that the Scilab federates can independently participate, synchronize and exchange data in the distributed interactive simulation using HLA through the RTI. The HLA services and the callback functions work properly.

6. Grid-enable Scilab and Distributed Interactive Simulation Grid

Similar techniques in integrating HLA with Scilab can also be applied to make Grid-enable Scilab. The advantages in using a Grid environment [12] are higher achievement in computing speed, saving in processing time, and greater simulation application and output.

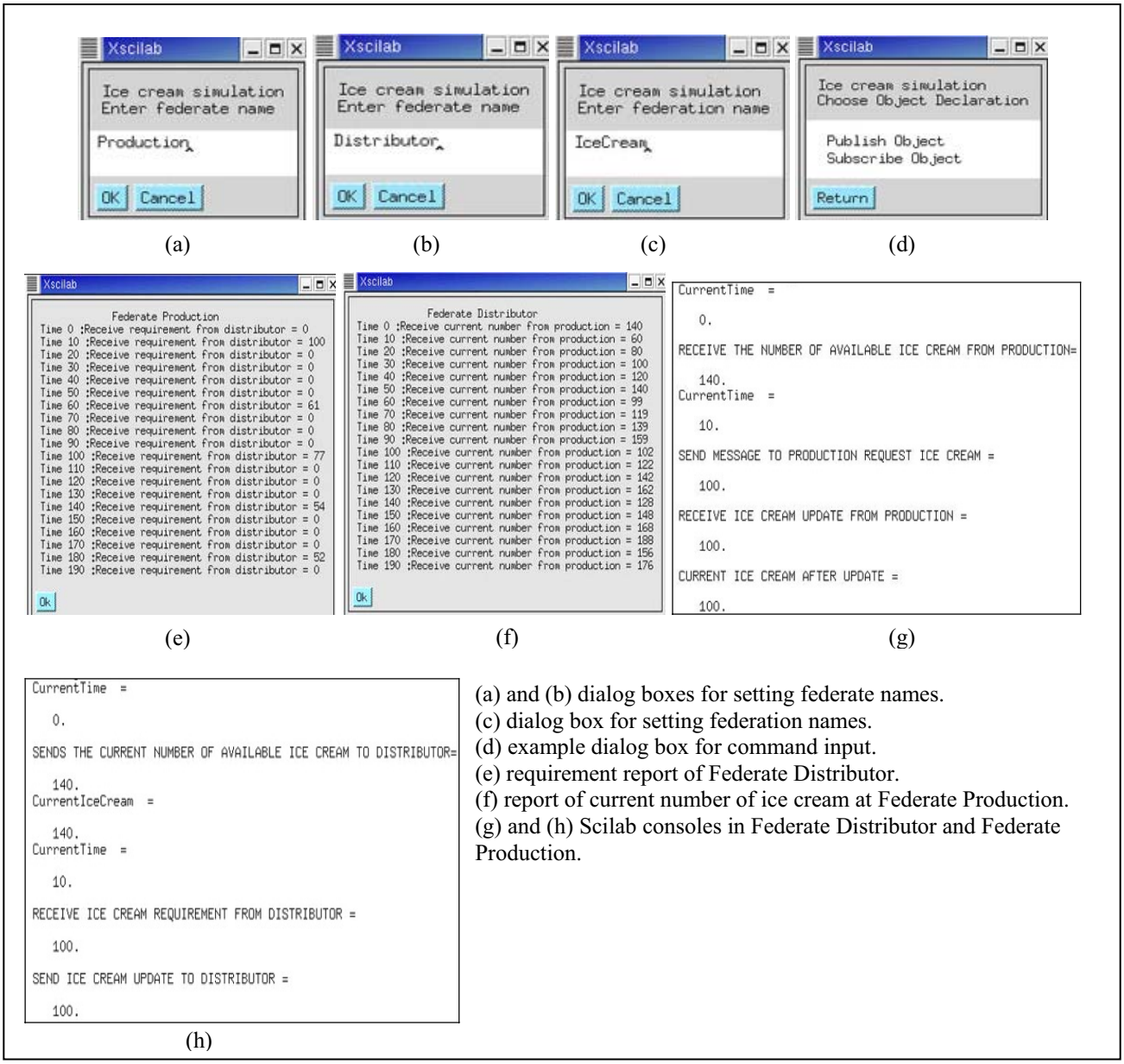
Current Grid Computing has not yet supported similar services to the distinguished HLA RTI distributed interactive simulation services including time management and ownership management. Although the RTI can be called by Grid participants to provide the services, as appeared in dsgrid [13], the HLA federation is considered a separate system. This situation is like building different transportation infrastructures, i.e., railway and bus systems that sometimes share the same junctions but run separately. It would be more efficient if Grid Computing includes such services.

Most researches about Grid Computing rather aim for enabling the dynamic "run-time" selection, sharing and aggregation of distributed autonomous resources. They hardly provide time synchronization or enable ownership transfer for distributed interactive simulation. The neglected issues are important to allow participation of distributed independent and interactive simulation nodes, especially in virtual reality and decision support systems. Consequently, this area is a real challenge to be explored.

7. Conclusion

This paper presents how to integrate the HLA with Scilab. An HLA module has been developed to enable Scilab to carry out distributed interactive simulations.

Basic RTI library services for distributed interactive simulation have been implemented and tested. Scilab federates can participate in distributed interactive simulations using HLA.



(a) and (b) dialog boxes for setting federate names.
(c) dialog box for setting federation names.
(d) example dialog box for command input.
(e) requirement report of Federate Distributor.
(f) report of current number of ice cream at Federate Production.
(g) and (h) Scilab consoles in Federate Distributor and Federate Production.

Figure 10. Example of the user interface.

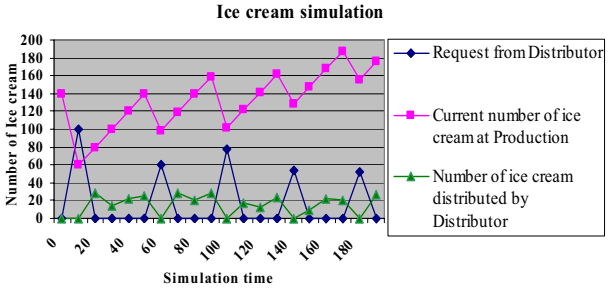


Figure 11. The result of the simulation.

This work is still ongoing. There are two more HLA services to be implemented and tested, including data distribution management and ownership management. Grid-enable Scilab and distributed interactive simulation Grid also appear to be interesting future works.

8. References

[1] Kuhl, F., R. Weatherly, and J. Dahmann, *Creating Computer Simulation Systems*, Prentice Hall PTR, 2000.

[2] S. Pawletta, B. Lampe, W. Drewelow, T. Pawletta, "HLA-based Simulation within an Interactive Engineering Environment", *Proceedings of 4th International Workshop on Distributed Simulation and Real Time Applications (DS-RT 2000)*, USA, 2000.

[3] Straβburger, S., *Distributed Simulation Based on the High Level Architecture in Civilian Application Domains*, ESV Verlagsgesellschaft mbH, Germany, 2001.

[4] Scilab Website, http://scilabsoft.inria.fr.

[5] Scilab Group INRIA Meta2 Project/ENPC Cergrene: "Scilab Internal", Guide for Developers, 1997. Available: ftp://ftp.inria.fr/INRIA/Scilab/documentation/pdf/.

[6] Institute of Electrical and Electronics Engineers (IEEE) Std 1516.1-2000, "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) Federate Interface Specification", 2001.

[7] MPI: "The Message Passing Interface", Available: http://www-unix.mcs.anl.gov/mpi.

[8] Institute of Electrical and Electronics Engineers (IEEE) Std 1516.1-2000, "Standard for Modeling and Simulation (M&S) High Level Architecture (HLA) Object Model Template (OMT) Specification", 2001.

[9] Defense Modeling and Simulation Office, "RTI Synopsis", RTI 1.3-Next Generation Programmer's Guide Version 5, 2002.

[10] Scilab Group INRIA Meta2 Project/ENPC Cergrene: "Interfacing C or Fortran programs with Scilab" Introduction to Scilab, 1998. Available: ftp:///ftp.inria.fr/INRIA/Scilab/documentation/pdf/.

[11] T. Theppaya, P. Tandayya, C. Jantaraprim, "Integrating HLA with Scilab", *Proceedings of the 2005 Fall Simulation Interoperability Workshop*, 2005.

[12] M. Baker, R. Buyya, D. Laforenza, "Grids and Grid technologies for wide-area distributed computing", Software-practice and Experience, DOI: 10.1002/spe.488, 2002.

[13] M. Lees, B. Logan, T. Oguara, and G. Theodoropoulos, "Simulating Agent-Based Systems with HLA: The case of SIM_AGENT -- Part II.", *Proceedings of the 2003 European Simulation Interoperability Workshop*, 2003.

Acknowledgement

This research is partly supported by the Thai National Grid Project.

Author Biographies

THITIMA THEPPAYA graduated in Computer Science from Prince of Songkla University (PSU), Thailand. Currently, she is pursuing a Master's degree in Computer Engineering at the Faculty of Engineering, PSU. Her thesis topic is *Development of a Distributed System of Scilab based on HLA.*

PICHAYA TANDAYYA graduated in Electrical Engineering (Communications) from Prince of Songkla University (PSU) in Thailand in 1990. She obtained her PhD in Computer Science in 2001 from the University of Manchester in the area of distributed interactive simulation. She is an Assistant Professor in the Department of Computer Engineering, PSU.

CHATCHAI JANTARAPRIM graduated in Electrical Engineering from Prince of Songkla University (PSU) in Computer Science in Thailand in 1991. He obtained his Master's degree in 2000 from the University of Manchester in the area of IC design. Currently, he is a lecturer in the Department of Computer Engineering, PSU.