Inter J Nav Archit Oc Engng (2012) 4:211~227
http://dx.doi.org/10.2478/IJNAOE-2013-0091

Implementation of the submarine diving simulation in a distributed environment

Sol Ha[1], Ju-Hwan Cha[2], Myung-Il Roh[3] and Kyu-Yeul Lee[4]

[1]*Department of the Naval Architecture and Ocean Engineering, Seoul National University, Seoul, Korea*
[2]*Department of Naval Architecture, Mokpo National University, Muan-Gun, Jeonnam, Korea*
[3]*School of Naval Architecture and Ocean Engineering, University of Ulsan, Mugeo-Dong, Ulsan, Korea*
[4]*Department of the Naval Architecture and Ocean Engineering, and Research Institute of Marine Systems Engineering, Seoul National University, Seoul, Korea*

ABSTRACT: *To implement a combined discrete event and discrete time simulation such as submarine diving simulation in a distributed environment, e.g., in the High Level Architecture (HLA)/Run-Time Infrastructure (RTI), a HLA interface, which can easily connect combined models with the HLA/RTI, was developed in this study. To verify the function and performance of the HLA interface, it was applied to the submarine dive scenario in a distributed environment, and the distributed simulation shows the same results as the stand-alone simulation. Finally, by adding a visualization model to the simulation and by editing this model, we can confirm that the HLA interface can provide user-friendly functions such as adding new model and editing a model.*

KEY WORDS: Submarine diving simulation; Distributed simulation; High Level Architecture (HLA) interface; Run-Time Infrastructure (RTI); Discrete EVent System specification (DEVS).

INTRODUCTION

Background of this study

A virtual object which acts as an actual object is called a 'simulation model' (simply, 'model') and the process which makes such object is called 'modeling'. A model consists of state variables, external and internal events, and state transition functions. As a set of instructions, a model needs some agent capable of actually executing the instructions and generating behavior and the agent is called a 'simulation engine'. To derive the process that status variables change by means of external and internal events, that is, to find the change of state variables according to the change of time is called 'simulation'. A simulation that the state of a model changes by means of any events is called a 'discrete event simulation'. The discrete event simulation processes the events, which change state variables of a model, in the order in which they occur. A simulation which calculates the state of a model every unit time is called a 'discrete time simulation'. The discrete time simulation is being mostly used for analyzing dynamics or mechanics systems because it calculates the state of a model every unit time.

A combined discrete event and discrete time simulation (simply, 'combined simulation') is a combined type of discrete event simulation and discrete time simulation. This simulation progresses by changing the state of a model according to the events. In addition, it progresses by calculating state variables of the model every unit discrete time when the state of the model is in the specific state. The model in this simulation come into being by a combination of the discrete event simulation model and the discrete time simulation model and, therefore, is called the combined discrete event and discrete time simulation

Corresponding author: *Myung-Il Roh*
e-mail: *miroh@ulsan.ac.kr*

model (simply, 'combined model'). The combined model is a standard formalism of modeling and simulation. The combined simulation is widely used in a stand-alone environment which is performed using one computer, but not in a distributed environment.

A distributed simulation means a simulation performed in distributed environment, which uses multiple computers at the same time. To simulate in a distributed environment, data distribution management and time management should be considered. The Defence Modeling and Simulation Office (DMSO) has suggested a standard specification, called High Level Architecture (HLA), which uses a distributed simulation middleware, called Run-Time Infrastructure (RTI) (DMSO, 1998a; DMSO, 1998b; DMSO, 1998c).

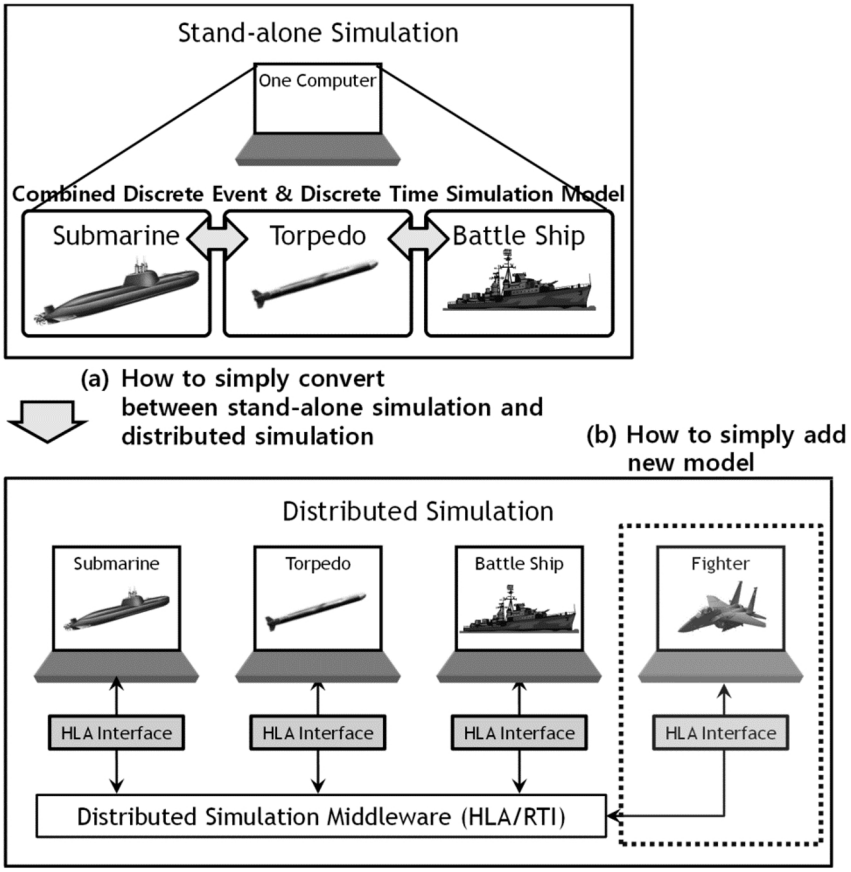


Fig. 1 Necessity of a HLA interface.

To use the combined model in a standard distributed environment, a HLA interface connecting the combined model with the HLA is required. Let's examine the necessity of this interface. Someone tried to change an existing stand-alone simulation of submarine battle scenario to a distributed simulation using the HLA/RTI. Because of the difference in the data structure between the combined model and the HLA/RTI, the interface connecting them are needed (Fig. 1(a)). In addition, using the interface, an existing 'fighter' model can easily participate in the distributed simulation (Fig. 1(b)) (Lee, 2006).

In this study, we developed a HLA interface to use the combined discrete event and discrete time simulation model in the distributed environment. The HLA interface is composed of an 'Interface Model', a 'Model Translator', and a 'HLA/RTI Translator'. The 'Interface Model', which is defined as a combined model, can be easily connected with combined models without any changes to them. The 'Model Translator' receives data from the 'Interface Model', translates data into the HLA/RTI functions, and transfers it to the HLA/RTI. The 'HLA/RTI Translator' receives data from the HLA/RTI, translates data into the combined model functions, and transfers it to the 'Interface Model'. A distributed simulation of 'diving submarine' was carried out by distributing submarine sub-systems into three simulation models and the HLA interface. It was confirmed that the distributed simulation using the HLA interface gives the same results as the stand-alone simulation. In addition, it was shown that the HLA interface gives user-friendly convenience such as adding an existing model and editing the model.

Related works

Some researches related to the HLA interface are being made in recent. Kuijpers, Lukkien, Brasse and Huijbrechts (1999) described and evaluated the Data Distribution Management (DDM) and Ownership Management (OM) services of the HLA interface for discrete time simulation. A distributed simulation model was constructed to measure the performance of the currently available DMSO RTI and to evaluate design trade-offs involving the use of these two management services. Lee and Zeigler (2005) reviewed existing message traffic reduction schemes and proposed a dynamic multiplexing approach as an efficient message traffic reduction scheme for discrete event simulation. This approach was based on a dynamic encoding of the joint output of sender components into a single message and was applicable to a model involving distributed components moving and interacting in multidimensional space by combining with predictive interest-based quantization. Zeigler, et al. (1998) described the Discrete EVent System specification (DEVS)/HLA distributed simulation environment under development and its support for predictive filtering research. The underlying DEVS and HLA frameworks were reviewed as a basis to discuss their synthesis in the form of a high level modeling and distributed simulation environment. Zeigler, Ball, Cho and Lee (1999) also presented the design and development of DEVS/HLA, an HLA-compliant modeling and simulation environment that supports high level model building using the DEVS methodology. Similarly, Lee, Zeigler and Venkatesan (2001) described the design and development of the DEVS/ Generic Data Distribution Management (GDDM) environment, a layered simulation environment that supports data distribution management.

Many researches related to the HLA interface have been made but most of them support discrete time simulation or discrete event simulation like DEVS, as mentioned above. To overcome these limitations, we proposed a HLA interface for combined discrete event and discrete time simulation model in distributed environment in this study.

The reminder of this paper is organized as follows: Section 2 gives background on the combined discrete event and discrete time simulation model and the distributed simulation. Section 3 gives detailed description of the developed HLA interface for the combined discrete event and discrete time simulation model, and Section 4 provides a distributed simulation for a diving submarine with the combined models and the HLA interface as the result of this study. Finally, Section 5 provides conclusions and directions for future work.

COMBINED SIMULATION MODEL AND DISTRIBUTED SIMULATION

The most important things to simulate in a distributed environment are a general purpose architecture and simulation middleware. The High Level Architecture (HLA) is a standard architecture for distributed computer simulation systems. Run-Time Infrastructure (RTI) is a middleware that is required when implementing HLA. For modeling and simulation, a combined discrete event and discrete time simulation model is widely used.

Combined discrete event and discrete time simulation model

Praehofer (1991) and Zeigler, Praehofer and Kim (2000) proposed a modeling and simulation method that can handle simulation models of a discrete event and discrete time, and it is widely used as a standard formalism of modeling and simulation. Using this model, a simulation engine progresses a simulation by changing the state of a model according to the events and by calculating state variables of the model every unit time when the state of the model is in the specific state. They also developed a simulation framework based on a proposed method with JavaBeans. Woo (2005) developed a simulation framework based on the commercial simulation tool called QUeuing Event Simulation Tool (QUEST) of Dassault System Co., Ltd. and applied it to shipbuilding process of a shipyard. He analyzed shipbuilding process in the viewpoints of production, process, and resource (PPR), and then defined simulation models for shipbuilding process by using ICAM DEFinition (IDEF) and Unified Modeling Language (UML). Bang (2006) and Cha, Roh and Lee (2010) developed a simulation framework based on combined model. However, the combined simulation of these studies is focused on a stand-alone simulation, so it cannot be used for a distributed simulation. To simulate in a distributed environment, simulation models need a standard architecture and simulation middleware. To solve this problem, that is, to implement the combined simulation in distributed environment, the standard architecture and simulation middleware, a HLA interface, was developed in this study. Of course, the concept of the simulation model and engine of this study is the same with that of Cha, Roh and Lee (2010).

High Level Architecture (HLA)

High Level Architecture (HLA) is a set of standard specifications for distributed simulations suggested by Defence Modelling & Simulation Office (DMSO) of Department of Defense (DoD) for data distribution and time management and interoperation between simulation models distributed on network (Lee, 2006). HLA provides the specifications of a common technical architecture for use across all classes of simulations. It provides the structural basis for simulation interoperability. The baseline definition of the HLA includes the HLA rules, the HLA interface specifications, and the HLA object model template. Using HLA, simulations can communicate with other computer simulations regardless of the computing platforms. Communication between simulations is managed by a Run-Time Infrastructure (RTI) (Lee, et al., 2006; Cha, et al., 2004a; Cha, et al., 2004b).

Run-Time Infrastructure (RTI)

The Run-Time Infrastructure (RTI) is a middleware that is required when implementing HLA and it is, in effect, a distributed operating system for the simulation model. The RTI provides a set of services that support the simulations in carrying out model-to-model interactions. All interactions among the models flow through the RTI. Using the HLA/RTI, Lee (Lee, 2006) developed the underwater vehicle's diving and surfacing simulation as a practical example of the underwater vehicle simulation.

HLA INTERFACE FOR COMBINED DISCRETE EVENT AND DISCRETE TIME SIMULATION MODEL

The strategy underlying the interface of a combined model to the HLA/RTI, illustrated in Fig. 2, is to exploit the informa-

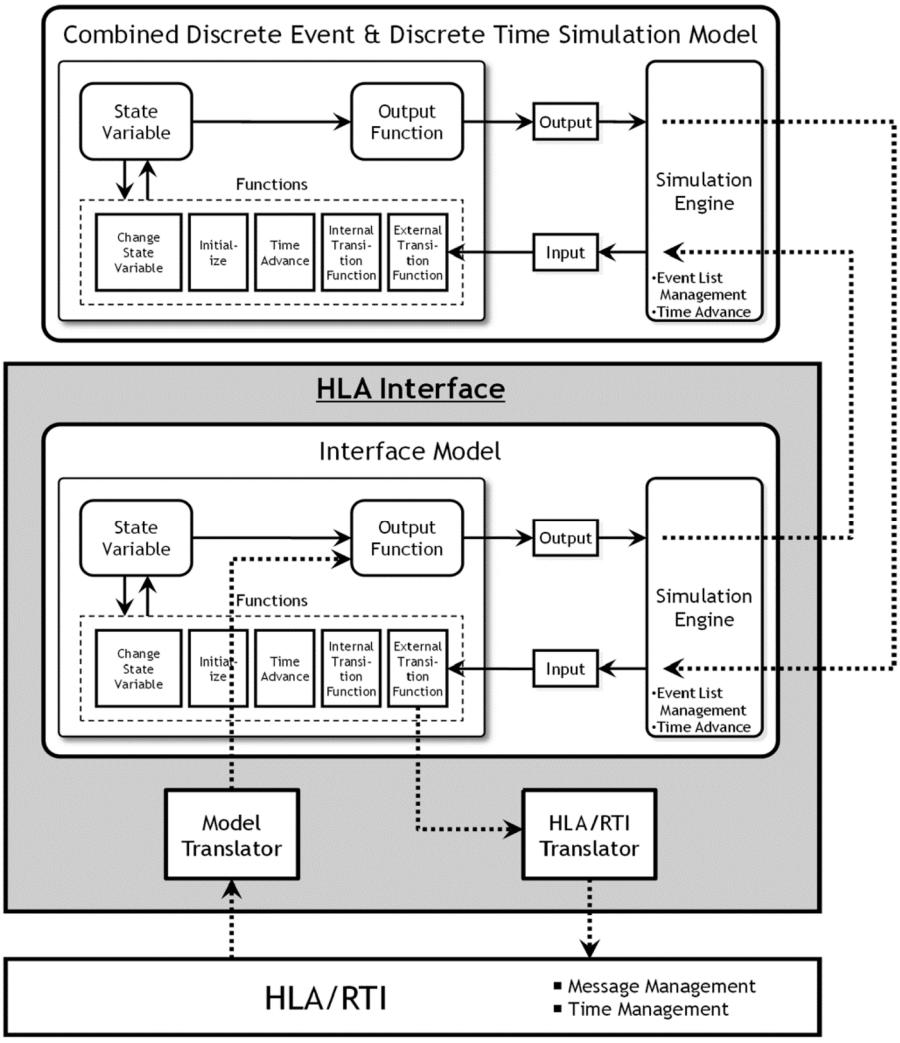


Fig. 2 Composition of a HLA interface.

tion contained within the combined model to automate as much as possible of the programming work required for constructing HLA compliant simulations. For example, the input and output of a combined model are automatically mapped into HLA interactions. Additionally, we seek to minimize any additional declarations to those that are minimally necessary for HLA operation. Our goals of the HLA interface are as follows:

(a) Can be applied to both a discrete event and discrete time.
(b) Can be used without changing the combined models.
(c) Can be used without additional compiling of the combined models.

For the specific purpose of not requiring changes to the simulation models, we defined the interface model as a combined model inside the HLA interface. Therefore, the HLA interface may exchange data with simulation models as if it were a stand-alone simulation using the combined model.

Combined discrete event and discrete time simulation

As shown in Fig. 3, the HLA interface is composed of an 'Interface Model' and two translators. The interface model is defined as a combined model. As intermediaries between the interface model and the HLA/RTI, the two translators communicate with the HLA/RTI.

Interface model

An existing model, which was used in a stand-alone simulation and will be connected with the HLA/RTI for a distributed simulation, is called a 'connected simulation model' in this study. The interface model is defined as a combined model. Because both a connected simulation model and the interface model are defined as combined models, the connected simulation model may exchange data with the interface model as if it were operated in a stand-alone simulation between two combined models. Therefore, the HLA interface can be used without any changes to the connected simulation models or additionnal compiling of them.

HLA/RTI translator

The HLA/RTI Translator translates functions of the interface model into HLA/RTI functions. It transfers the data, which is generated on the output function of interface model, to the HLA/RTI.

Model translator

The Model Translator translates HLA/RTI functions into functions of the interface model. It transfers the data, which the HLA/RTI sends, to the output function of the interface model.

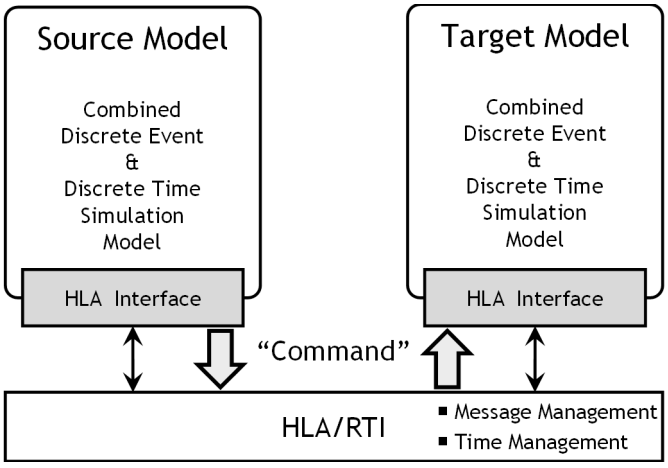


Fig. 3 Process of transferring the data using the HLA interface.

Process of the HLA interface

Using the HLA interface, the process of transferring the data between source model and target model can be divided into two steps: (The model which sends the data is called a 'Source Model' and the model which receives the data is called a 'Target Model'.)

(a) Step 1: transferring data from a simulation model to the HLA/RTI (Source Model → Interface Model → HLA/RTI Translator → HLA/RTI).
(b) Step 2: transferring data from the HLA/RTI to a simulation model (HLA/RTI → Model Translator → Interface Model → Target Model).

Fig. 3 is an example of how simulation models transfer data using the HLA interface. As shown in Fig. 3, the source model transfers the data "Command" to the target model.

Process of transferring data from source model to the HLA/RTI

Using the HLA interface, the process of transferring the data from the source model to the HLA/RTI is as follows:
(a) The source model generates a discrete event, "Command" (Fig. 4(a)).
(b) Using the output function of the source model, the "Command" event is transferred to the simulation engine of the source model (Fig. 4(b)).

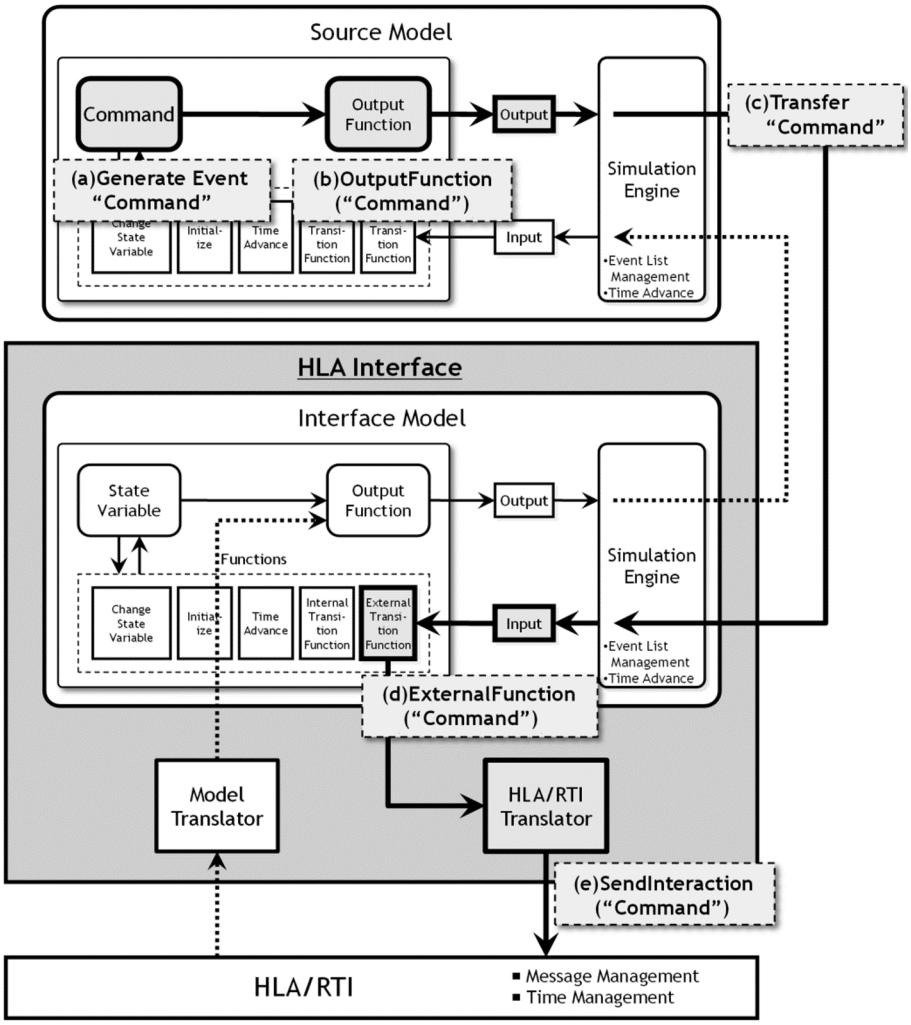


Fig. 4 Simulation using the HLA interface-process of transferring data from the source model to the HLA/RTI.

(c) The simulation engine of the source model transfers the "Command" event to the interface model of the HLA interface. The simulation engine of the interface model enters the "Command" event into the external transition function of the interface model (Fig. 4(c)).

(d) Using the external transition function of the interface model, the interface model transfers the "Command" event to the HLA/RTI translator of the HLA interface (Fig. 4(d)).

(e) The HLA/RTI translator translates the external transition function of the interface model into a HLA/RTI function. That is, the function ExternalFunction ("Command") is translated to the function SendInteraction ("Command"). Then, the HLA/RTI translator sends the translated function to the HLA/RTI (Fig. 4(e)).

Process of transferring the data from the HLA/RTI to the target model

Using the HLA interface, the process of transferring the data from the HLA/RTI to the target model is as follows:

(a) Using the ReceiveInteraction ("Command") function, which is a kind of HLA/RTI function, the HLA/RTI sends the "Command" event to the model translator of the HLA interface (Fig. 5(a)).

(b) The model translator translates the HLA/RTI function into an output function of the interface model. That is, the function ReceiveInteraction ("Command") is translated into the function OutputFunction ("Command"). Using the output function, the interface model transfers the "Command" event to the simulation engine of the interface model (Fig. 5(b)).

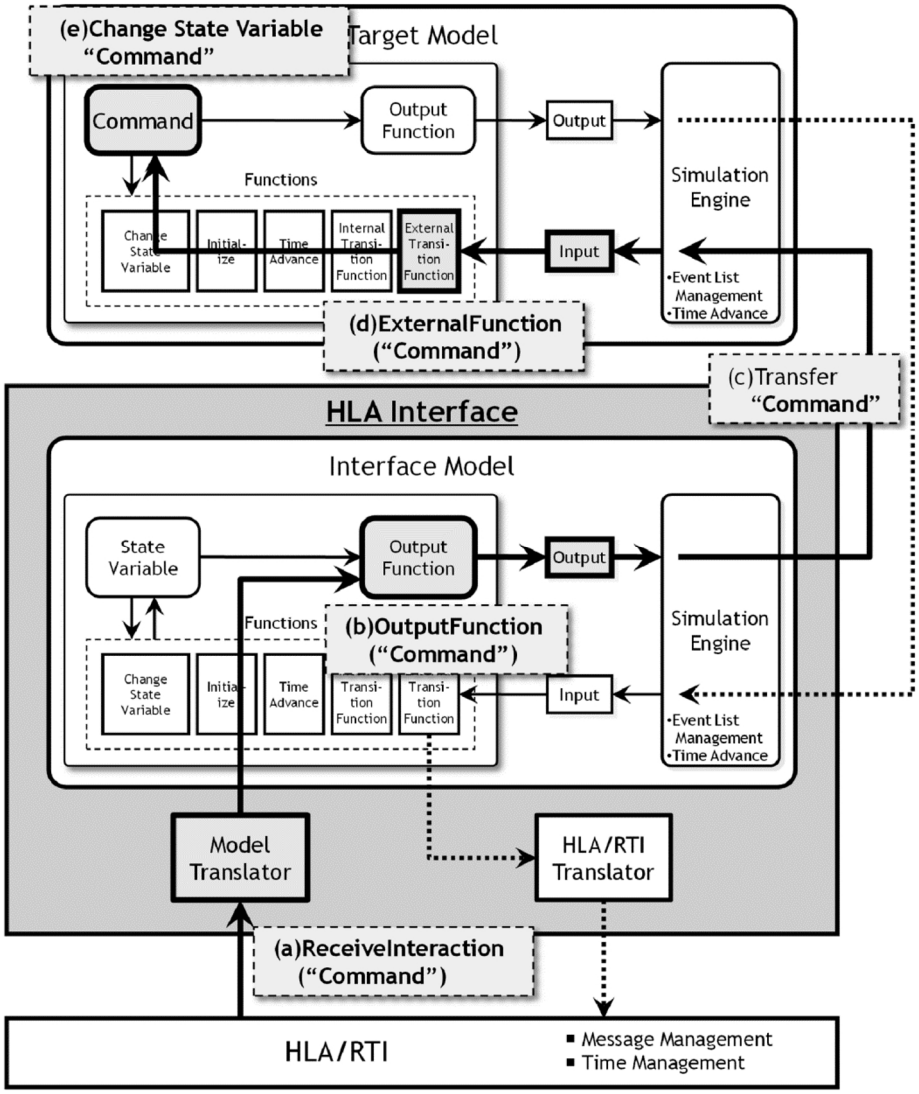


Fig. 5 Simulation using the HLA interface-process of transferring the data from the HLA/RTI to the target model.

(c) The simulation engine of the interface model sends the "Command" event to the simulation engine of the target model. The simulation engine of the target model enters the "Command" event into the external transition function of the target model (Fig. 5(c)).
(d) Using the external transition function, the target model changes a state variable to the "Command" state (Fig. 5(d), (e)).

Time management of the HLA interface

Fig. 6 shows an example of time management of the HLA interface between Model 1 and Model 2. First, when the simulation starts, the Model 1 sends TimeAdvance (1) to the Model 1 interface which means there is no output event until 1 *sec*, and the Model 1 interface sends NextMessageRequestAvailable (1) to the RTI which asks for a message of the time of 1 *sec* or less. At the same time, the Model 2 sends TimeAdvance (10) to the Model 2 interface which means there is no output event until 10 *sec*, and the Model 2 interface sends NextMessageRequestAvailable (10) to the RTI which asks for a message of the time of 10 *sec* or less. The RTI progresses the earliest time after it receives NextMessageRequestAvailable from all models. That is, the RTI sends TimeAdvanceGrant (1) to the Model 1 interface and the Model 2 interface which means there is no message of the time of 1 *sec* or less. Now, the Model 1 and the Model 2 progresses the simulation to 1 *sec*.

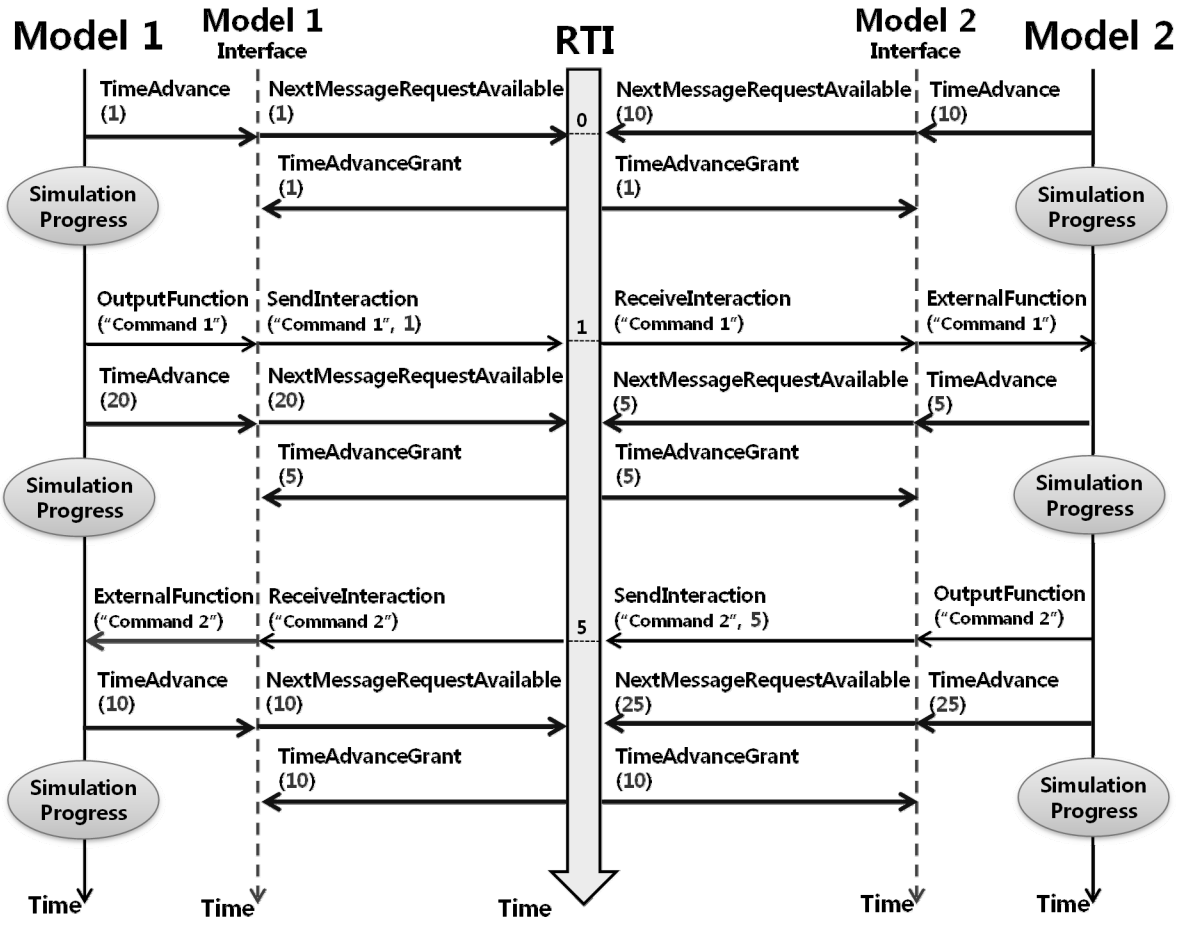


Fig. 6 Example of time management of HLA interface between Model 1 and Model 2.

If the Model 1 sends "Command 1" event to the Model 1 interface using an output function called OutputFunction ("Command 1"), the Model 1 interface translates the event into SendInteraction ("Command 1", 1) and sends the "Command 1" event to the RTI at 1 *sec*. Then, the RTI sends the "Command 1" event to the Model 2 interface using the RTI function called ReceiveInteraction ("Command 1"), and the Model 2 receives the "Command 1" event using the external transient function called ExternalFunction ("Command 1"). Using the output function, the Model 1 sends TimeAdvance (20) to the Model 1 interface which means there is no output event until 20 *sec*, and the Model 1 interface sends NextMessageRequestAvailable (20)

to the RTI. After performing the external event, the Model 2 sends TimeAdvance (5) to the Model 2 interface which means there is no output event until 5 *sec*, and the Model 2 interface sends NextMessageRequestAvailable (5) to the RTI.

The RTI sends TimeAdvanceGrant (5) to the Model 1 interface and the Model 2 interface to progress the simulation to 5 *sec* which is the earliest time, and then the Model 1 and the Model 2 progress the simulation to 5 *sec*. The simulation time is synchronized and progresses in the form that the "Command 2" event is sent to the Model 1 through the Model 2 and the RTI at 5 *sec*.

Usage of the HLA interface

Fig. 7 shows the usage of the HLA interface. The HLA interface enables a simulation model, used in a stand-alone simulation, to be connected to the HLA/RTI without any changes.

(a) Create an instance of the simulation model which is used in a stand-alone simulation (Fig. 7(a)).
(b) Create an instance of the HLA interface and connect it with the instance of the simulation model (Fig. 7(b)).
(c) Start a simulation. The following information is required to start the simulation (Fig. 7(c)):
- The name of the simulation.
- The name of the file which has information on the inputs and outputs of the simulation model.
- The name of the simulation model.
- The name of other simulation models participating in the simulation.
(d) At the end of the simulation, delete the instance of the HLA interface (Fig. 7(d)).

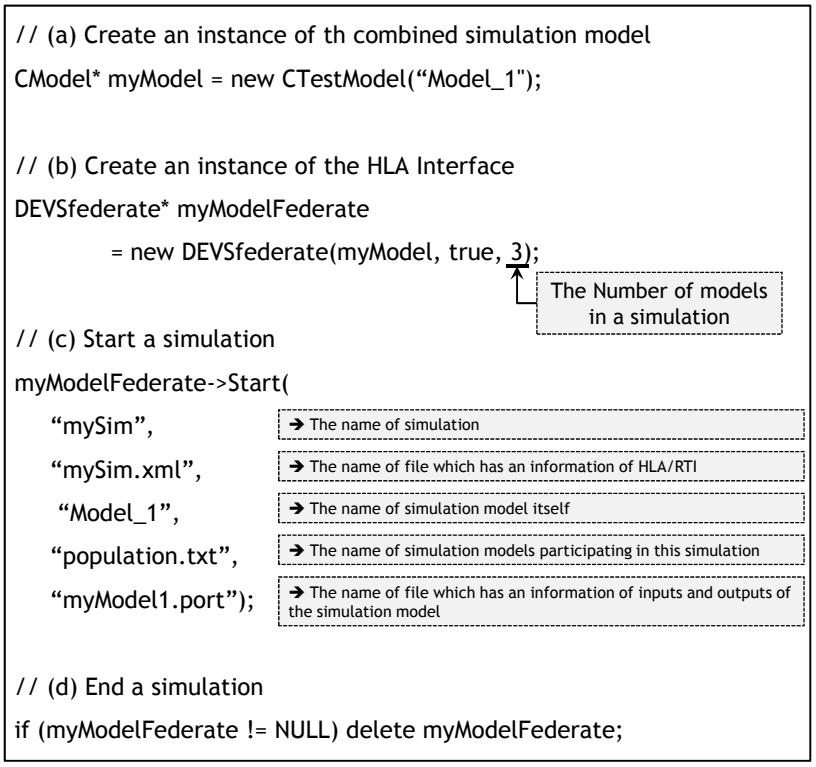


Fig. 7 Usage of the HLA interface.

The name of simulation models participating in the simulation-we call this 'population information'-should be given with a text file "population.txt". The text file should contain the number of simulation models and the name of each simulation model participating in the simulation (Fig. 8(a)). The input and output information of each simulation model should be described in each port file (Fig. 8(b)).

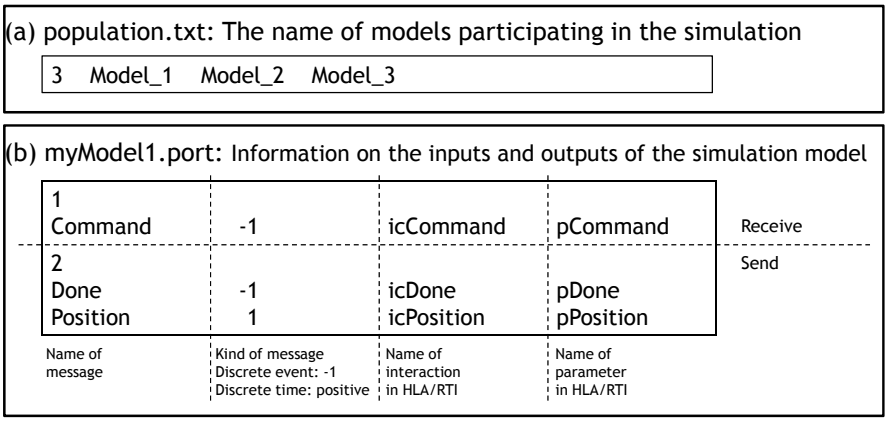


Fig. 8 Names and input/output information of models in a distributed simulation.

Conversion to a distributed simulation and the addition of other models using the HLA interface

Using the HLA interface, we can simply convert a stand-alone simulation into a distributed simulation. Fig. 9 shows that a stand-alone simulation used on a submarine system can be converted into a distributed simulation. As described in Section 3.1, the submarine simulation model defined as combined model can be used in a distributed simulation without any changes to the model.

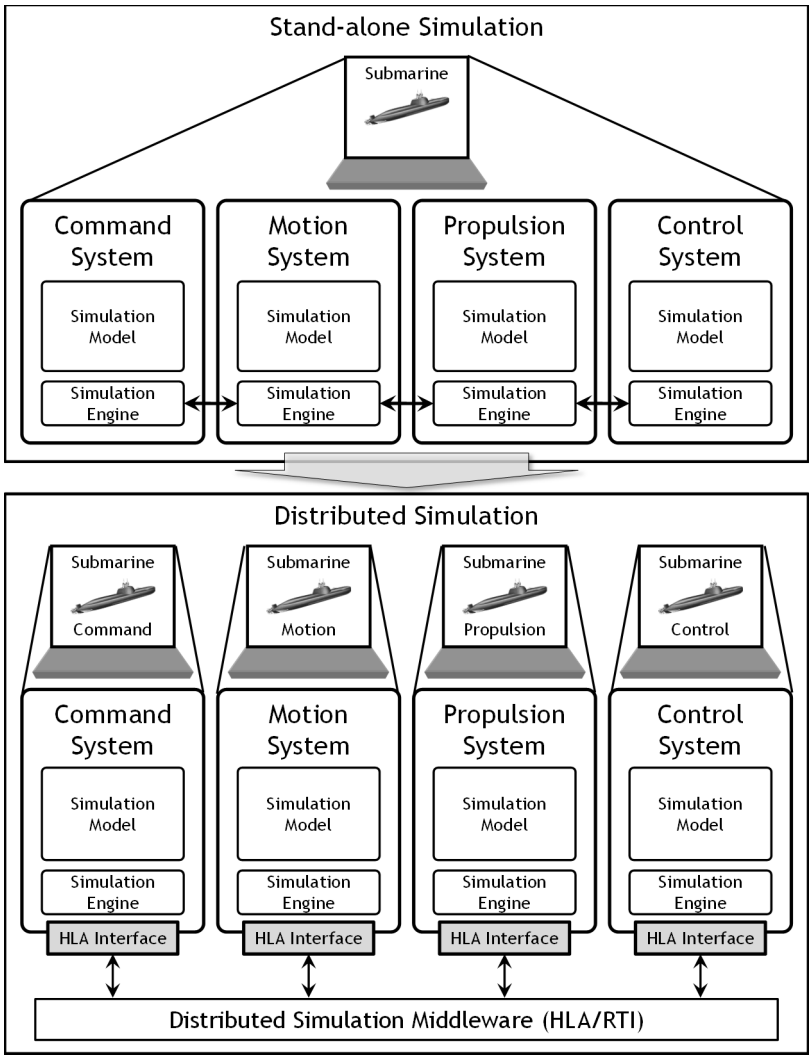


Fig. 9 Conversion from stand-alone simulation to distributed simulation.

In addition, it is very simple to add another simulation model to a distributed simulation. As shown in Fig. 10, to add a new visualization model to the existing distributed simulation, it is only needed to define the visualization model as a combined model. Then, the model can be connected to the HLA/RTI using the HLA interface. Finally, the simulation information such as the population information and each port file should be modified.

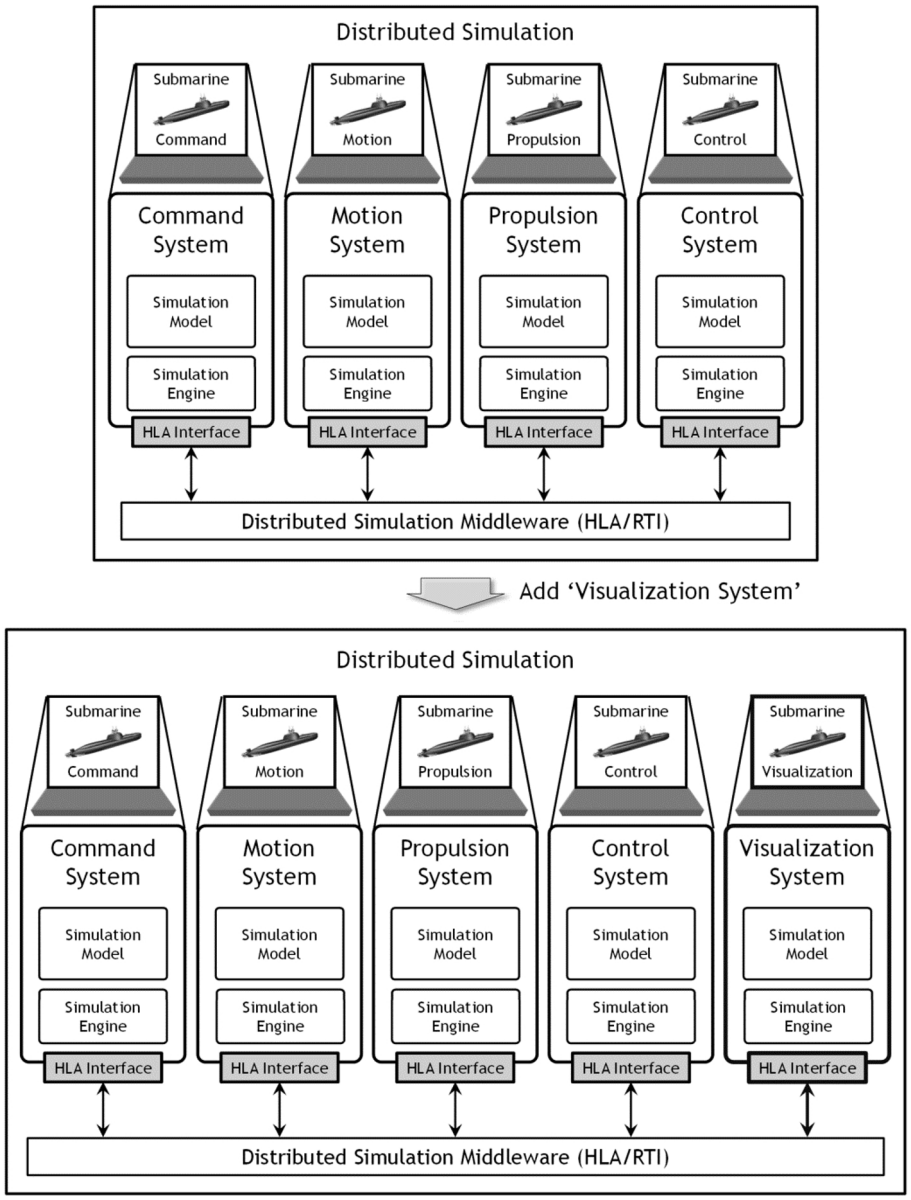


Fig. 10 Addition of another simulation model using the HLA interface.

SUBMARINE DIVING SIMULATION IN A DISTRIBUTED ENVIRONMENT

To verify the function of the HLA interface, a submarine diving scenario was simulated in a distributed environment and the simulation results were compared with a stand-alone simulation. First, the submarine diving scenario is defined.

Conversion to a distributed simulation and the addition of other models using the HLA interface

The submarine diving scenario is as follows:

(a) Stop the diesel engine and set the power source to the battery.

(b) Close all air-valves and the hatch to block out the outside air.
(c) Fill the tanks with seawater by opening the vents in the main ballast tank, the compensate tank, and the negative tank.
(d) Using the rudders, keep a trim angle of about 6~8 *degrees*.
(e) To keep neutral buoyancy, empty the negative tank 15 m before the target depth.
(f) To exhaust the air in the main ballast tank, perform leveling.
(g) Keep neutral buoyancy using the compensate tank and maintain the trim angle by using the trim tank.
(h) Dive to the target depth using the rudders.

Based on this scenario, we devised simulation models for the submarine diving simulation.

Submarine diving simulation using a combined model

A submarine is composed of many systems and sub-systems such as a command system, a motion system, a propulsion system and a control system. The submarine diving scenario, described in Section 4.1, is discrete events in a simulation and it is a part of the command system. Each process in this scenario requires discrete time simulation such as:

- The submarine follows a path in the water (Motion System).
- The submarine follows a target path using rudders and ballast tanks (Control System).

Ham, et al. (2008) and Son, et al. (2008) defined four sub-systems within the submarine's systems; the command system, the motion system, the propulsion system, and the control system. The command system, which contains many events during diving, can be defined as a discrete event simulation model. The motion system, propulsion system, and control system can be defined as discrete time simulation models because they are based on differential equations of the submarine's dynamics. The motion system calculates the position and attitude of the submarine using 6 degree-of-freedom dynamics (Gertler and Hagen, 1967). The propulsion system calculates external forces such as a fluid force, a buoyancy force, and a gravity force. The control system calculates the control force, which is generated by the control planes (Fig. 11).

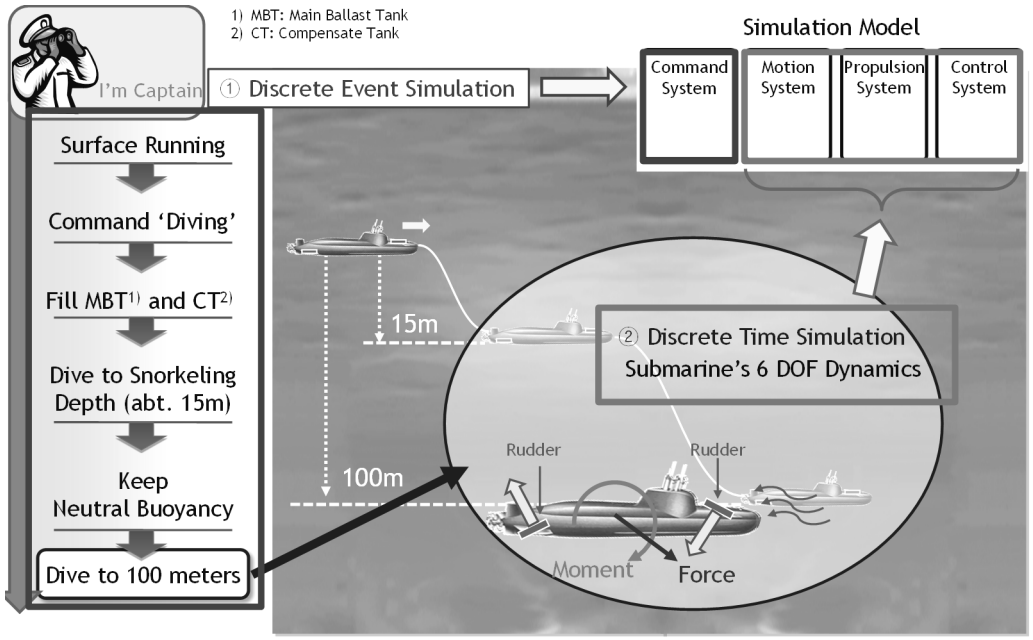


Fig. 11 Submarine diving simulation using a combined model.

Fig. 12 shows the configuration of the submarine diving simulation models. Each sub-system is defined as the combined model, and exchanges the data for the submarine diving simulation.

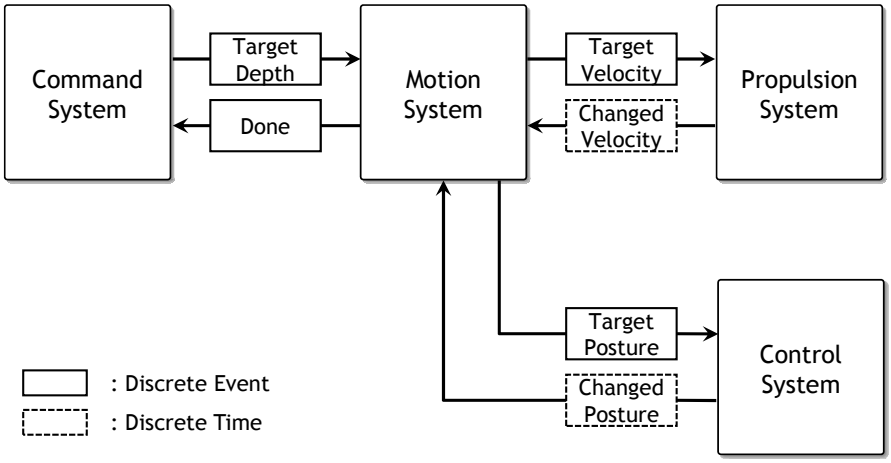


Fig. 12 Configuration of the submarine diving simulation model (a stand-alone simulation).

Distributed simulation using HLA interface

We configured a stand-alone simulation of a diving submarine in Section 4.2. In this section, the distributed simulation of the diving submarine was constructed. The HLA interface is used to connect to each of the submarine's systems through the HLA/RTI (Fig. 13).

Using discrete event simulation, the diving scenario is simulated as described in Section 4.1. Initially, the submarine sails the sea-surface at a speed of 10 *knots*. At the start of diving, the main ballast tank is filled with seawater so the submarine lists to the fore. To keep the trim angle at 6 *degrees*, the control planes are used to control the attitude of the submarine. The submarine was set to dive to 15 *meters*, and then to 100 *meters*.

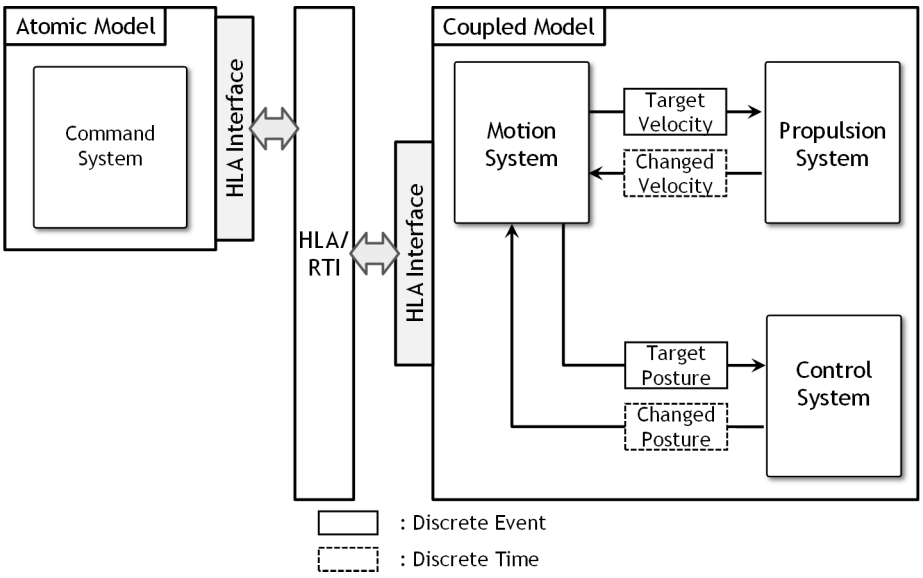
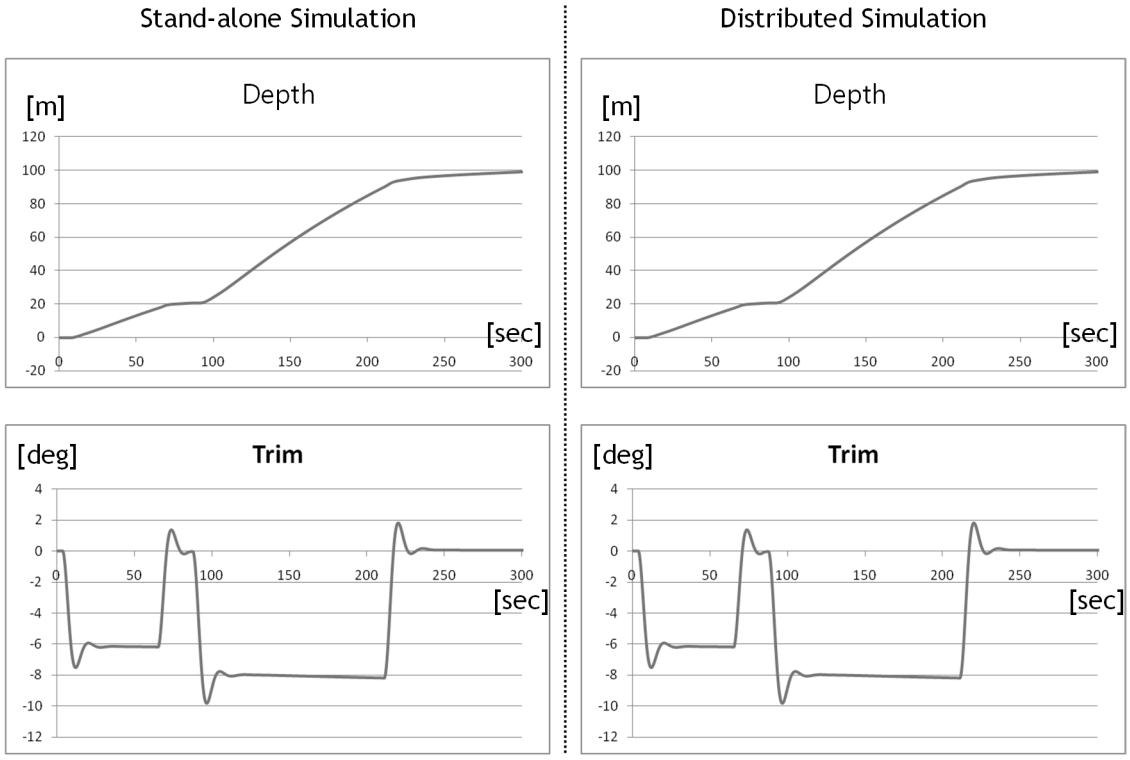


Fig. 13 Configuration of submarine diving simulation model using the HLA interface (a distributed simulation).

Comparison of results between the distributed simulation and the stand-alone simulation

In Section 4.2, the submarine diving simulation was carried out on a single computer, whereas the same simulation was performed on multiple computers as described in Section 4.3. In both simulations it can be seen that the depth and trim angle of the submarine were the same at each unit of time (Fig. 14). In this simulation, the submarine was set to dive to a depth of 15 *meters*, and then to a depth of 100 *meters*. The trim angle of the submarine was kept at 6 *degrees* to the depth of 20 *meters*, and 8 *degrees* to the depth of 100 *meters*.



(a) (b)

Fig. 14 Comparison of the depth and trim angle of the submarine in the stand-alone simulation (a) and the distributed simulation (b).

In the distributed simulation, it takes longer time than the stand-alone simulation because the distributed simulation needs the time for translating events or commands and then translating them to other computers through the network as compare to the stand-alone simulation. As shown in Table 1, in the case of the stand-alone simulation which was carried out on a single computer, it took about 221.472 *sec*. However, in the case of the distributed simulation which was carried out on multiple computers, it took about 631.295 *sec*.

Table 1 Comparison of the computation time for submarine diving simulation between the stand-alone simulation and the distributed simulation.

Simulation type	Computation time (*sec*)
Stand-alone simulation	221.472
Distributed simulation	631.295

Adding and editing another simulation model using the HLA interface

Using the HLA interface, it is very simple to add another simulation model to a distributed simulation. To visualize the position and attitude of the submarine from the motion system during the submarine diving simulation, a visualization system was added.

First of all, the visualization system is defined as the combined model and connected with the HLA/RTI using the HLA interface (Fig. 15). The visualization system exchanges the submarine's position and attitude with the motion system (Fig. 15). To connect with the HLA/RTI, the simulation information like input and output of the simulation model was modified (see Section 3.3). Fig. 16 shows that the visualization system performed visualization of the submarine's position and attitude.

Then, to visualize the rudder angle of the submarine additionally, the port file which contains the input and output information of each model was simply modified (Fig. 17). Fig. 18 shows the visualization of the rudder angle of the submarine. As

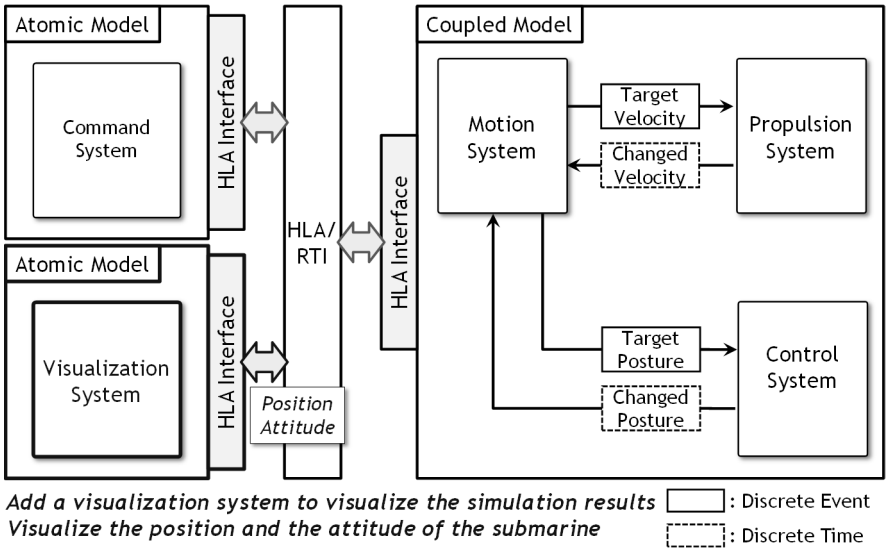


Fig. 15 Adding of a visualization system.

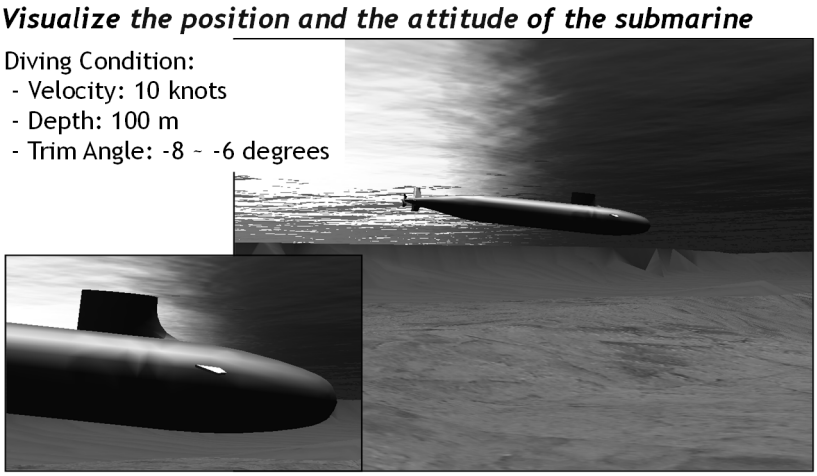


Fig. 16 Composition of a HLA interface.

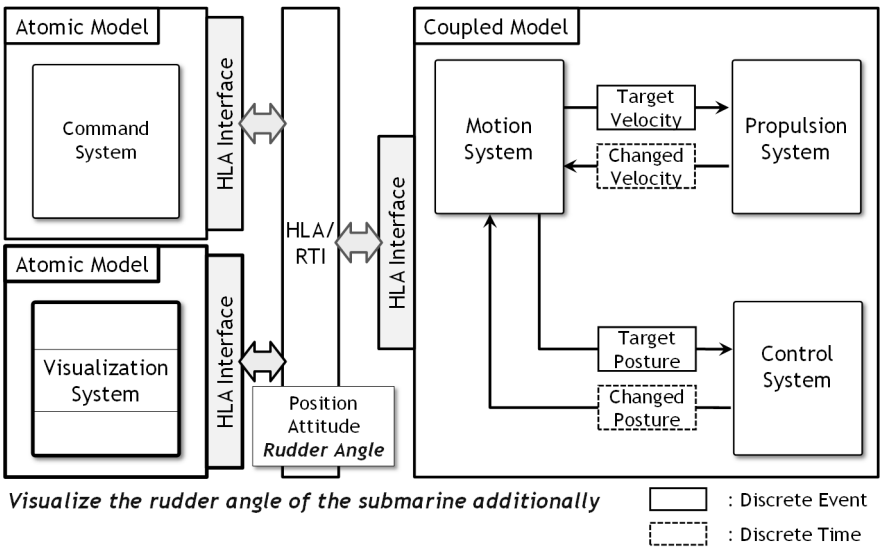


Fig. 17 Editing a visualization system-visualize the rudder angle of the submarine additionally.

this example, the combined model can be easily added to the distributed simulation and also the simulation information of the models can be simply edited by using the HLA interface.

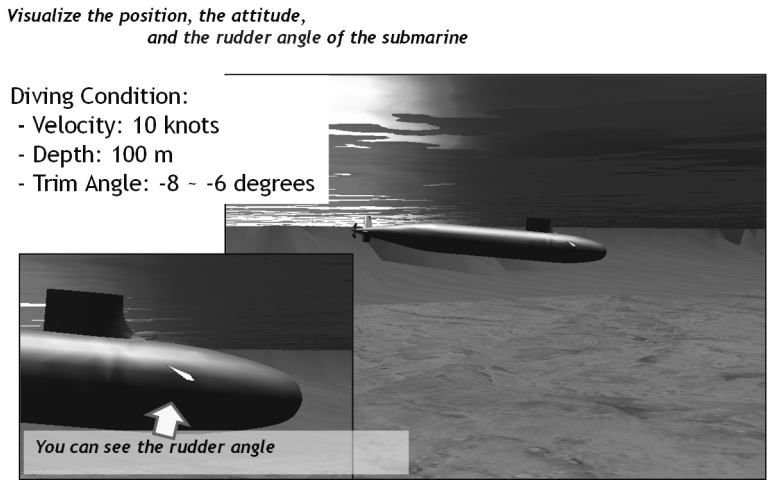


Fig. 18 Change of the position, the attitude, and the rudder angle of the submarine.

CONCLUSION AND FUTURE WORK

In this study, we developed a HLA interface which can easily connect combined discrete event and discrete time simulation models with the HLA/RTI. The HLA interface has an interface model, model translator, and HLA/RTI translator. For simple connections with a combine model, the interface model is defined as a combined model. Two translators help the interface model to communicate with the HLA/RTI. With the submarine diving scenario, it was proven that the distributed simulation, which uses the HLA interface, gives the same results as the stand-alone simulation. In addition, it was confirmed that the HLA interface provides user-friendly functions such as adding new models and editing them. Future work will be to perform the verification and validation of the HLA interface with more examples. In addition, the developed interface will be applied to more complicated example which we can verify the advantage of the distributed simulation.

ACKNOWLEDGEMENTS

This work was supported by:

(a) Underwater Vehicle Research Center (SM-11: "A Study on the Network-based Architecture of Virtual System for the Simulation of Underwater Vehicles").

(b) Industrial Strategic Technology Development Program (10035331, Simulation-based Manufacturing Technology for Ships and Offshore Plants) funded by the Ministry of Knowledge Economy, Korea.

(c) Research Institute of Marine System Engineering of Seoul National University.

(d) Marine Technology Education and Research Center by Brain Korea 21 of Seoul National University.

(e) National Research Foundation of Korea Grant funded by the Korean Government (R33-2008-000-10150-0).

REFERENCES

Bang, K.W., 2006. *Combined discrete event and discrete time simulation framework for shipbuilding process planning*. M. Sc. Seoul National University.

Cha, J.H., Lee, S.J., Yoo, S.J., Lee, K.Y., Choi, H.S. and Seong, W.J., 2004a. A study on the distributed simulation of the underwater vehicle using the high level architecture. *Proceeding of the society of naval architecture of Korea*. Jeju, Korea, 22-23 April 2004. pp.1494-1505.

Cha, J.H., Lee, S.J., Yoo, S.J., Lee, K.Y., Choi, H.S. and Seong, W.J., 2004b. A study on the modeling and simulation based on the HLA (High Level Architecture) for developing the virtual prototype technique of the underwater vehicle. *Proceeding of the society of CAD/CAM engineers in Korea*. Pyeongchang, Korea, 12-14 February 2004. pp.345-354.

Cha, J.H., Roh, M.I. and Lee, K.Y., 2010. Combined discrete event and discrete time simulation framework and its application to the block erection process in shipbuilding. *Advances In Engineering Software*, 41(4), pp.656-665.

DMSO, 1998a. *HLA Rules version 1.3*. IEEE Standards Draft.

DMSO,1998b. *HLA Interface Specification version 1.3*. IEEE Standards Draft.

DMSO, 1998c. *HLA Object Model Template Specification version 1.3*. IEEE Standards Draft.

Gertler, M. and Hagen, G.R., 1967. *Standard equations of motion for submarine simulation*. Naval Ship Research and Development Center.

Ham, S.H., Cha, J.H., Lee, K.Y., Park, K.P. and Roh, M.I., 2008. Submarine diving and surfacing simulation using discrete event and dynamic-based discrete time combined modeling architecture. *Proceedings of the society of naval architecture of Korea*. Changwon, Korea, 13-14 November 2008. pp.163-171.

Kuijpers, N.H.L., Lukkien, J.J., Brasse, M.H.H. and Huijbrechts, S., 1999. Applying data distribution management and ownership management services of the HLA interface specification. *Proceedings of simulation interoperability workshop*. Orlando, USA, 12-17 September 1999. 99F-SIW-023.

Lee, J.S. and Zeigler, B.P., 2005. Dynamic multiplexing and high-performance meling in dstributed simulation. *Journal of Simulation*, 81(5), pp.365-380.

Lee, J.S., Zeigler, B.P. and Venkatesan, S.M., 2001. Design and development of data distribution management environment. *Journal of Simulation*, 77(1-2), PP.39-52.

Lee, S.J., 2006. *A study on model structure based on high level architecture for distributed simulation of underwater vehicles*. Ph.D. Seoul National University.

Lee, S.J., Cho, D.Y., Kang, J.H., Lee, H.K., Lee, K.Y., Kim, T.W., Han, S.H. and Jung, H.S., 2006. A study on the model structure based on the high level architecture for the distributed simulation of the underwater vehicles. *Proceedings of the society of CAD/CAM engineers in Korea*. Pyeongchang, Korea, 9-10 February 2006. pp.913-921.

Praehofer, H., 1991. *System theoretic foundations for combined discrete-continuous system simulation*. Ph.D. Thesis. Johannes Kepler University. Linz, Austria.

Son, M.J., Lee, H.J., Ham, S.H., Lee, H.K., Cho, D.Y., Roh, M.I., Kim, T.W., Lee, K.Y., Han, S.H. and Nah, Y.I., 2008. Development of submarine simulation model architecture using combined discrete event and discrete time system specification. *Proceeding of the society of naval architecture of Korea*. Jeju, Korea, 29-30 May 2008. pp.1659-1668.

Woo, J.H., 2005. *Modeling and simulation of indoor shop system of shipbuilding by integration of the product, process, resource and schedule information*. Ph.D. Seoul National University.

Zeigler, B.P., Ball, G., Cho, H., Lee, J.S. and Sarjoughian, H., 1998. *The DEVS/HLA distributed simulation environment and its support for predictive filtering*. Technical Report, University of Arizona, Tucson, USA.

Zeigler, B.P., Ball, G., Cho, H. and Lee, J.S., 1999. Implementation of the DEVS formalism over the HLA/RTI: Problems and solutions. *Proceedings of Simulation Interoperation Workshop.* Orlando, USA, 14-19 March 1999.

Zeigler, B.P., Praehofer, H. and Kim, T.G., 2000. *Theory of modeling and simulation*. 2nd Ed. New York: Academic Press.