Linking Discrete Event Simulation Models Using HLA

Darren J. Price
Intelligent Systems Research Lab
Deakin University
Victoria, Australia
djp@deakin.edu.au

Saeid Nahavandi
Intelligent Systems Research Lab
Deakin University
Victoria, Australia
nahavand@deakin.edu.au

Simon Walsh and Doug Creighton
Intelligent Systems Research Lab
Deakin University
Victoria, Australia
(spwalsh, dougc)@deakin.edu.au

Abstract – *The increasing usage of discrete-event simulation packages for modeling and analyzing manufacturing and logistics has led to a need for connecting simulation models together at runtime. One such methodology for linking discrete-event simulation models together has been developed for this research and this paper demonstrates the usage of this linking method. A unified simulation model is developed from two sub-models developed using different simulation packages.*

Keywords: High Level Architecture, Digital Factory, Distributed Simulation, Unified Simulation Model.

1 Introduction

With more and more manufacturing industries using discrete event simulation to help solve and reduce their problems, there is an increasing need for linking different simulation packages as there are more COTS (Commercially available Off the Shelf) simulation packages available. As large manufacturing industries move toward using simulation and modeling and change from using older simulation packages to newer techniques, it has been seen that they use more than one COTS simulation package throughout a factory. When different sections of a factory are modeled separately, using either the same package (making different models), or a different package, a series of 'simulation islands' can appear [1]. While these separate models may accurately model their respective areas they tend to not help gain a complete view of the factory in progress. A model of a large factory is easier to create and maintain if created in several smaller parts [6][7]. Therefore, this research has aimed to investigate and develop a method by which models of manufacturing systems can be created in different COTS simulation packages and interfaced with one another at runtime.

2 Why Unify Simulation Models

There are several reasons why unifying several smaller simulation models at runtime rather than developing a single large model is advantageous. Firstly, modeling a system as several small parts allows multiple people to work on developing the models. Existing discrete event simulation packages are generally not suited to having multiple people working on the one model. A second reason is the situation where models have previously been made for some sections of the manufacturing plant or supply chain. In this situation the question arises whether it is better to attempt to make the existing models reusable or is it better to start from scratch and remodel the entire factory again, all in one simulation package. Allowing existing models to be linked can greatly save time and resources. These simulations may (or may not) be simulated in one or more simulation packages, which means, to get a greater view and understanding of what is happening at factory level, the islands need to communicate and be linked together [1]. A third reason for linking models at runtime is the situation where separate companies (or other such entities) do not wish to share their information with each other. If they were to develop models of their operations independently of each other to be linked at runtime their information would be secure.

At present it is very difficult to combine different discrete event models together that have been modeled using different simulation packages, or even with the same simulation package. This is because no standard language and/or interface method for discrete event simulation software has yet been developed. Some researchers have begun working on this, for example, Hibino connected different simulators such as Quest, Simple++ and Garops [8][9][10] together. This was done by using the HLA architecture as the IEEE standard 1516 [5] and a commercially available HLA/RTI (pitch) and then creating a manufacturing adapter to communicate between the HLA/RTI and the simulation software. This idea has been extended on for this research with the creation of a wrapper for both Quest and Arena [11] and a demonstration of this system in action is the focus of this paper.

3 High Level Architecture (HLA)

HLA evolved from a U.S. Department of Defense (DoD) project and its roots came from a military simulation, making defense the main area HLA is used at this current stage [2][3][4]. Although HLA is originally a U.S. DoD development, (now an IEEE standard 1516 [5]) the U.S. DoD are encouraging other areas of simulation to take advantage of the new development. Not only does the U.S.

DoD encourage other areas such as manufacturing, supply chain management, automotive, airport and even health care to use HLA, they also offer some support with development that other areas of simulation can benefit from [2]. The HLA protocol has taken large steps towards increasing the interoperability of simulations, however when connecting COTS it only offers a partial solution to the problem; because the original HLA Runtime Infrastructure (RTI) was designed by the U.S. DoD for custom simulations, it does not have any easy way to connect two or more COTS together. Only recently has the HLA/RTI started to branch from military applications to other areas to which it may be applied.

HLA states that each sub-model becomes a federate. These federates are connected to create a federation (or larger model made from smaller sub-models). The RTI is the program that is responsible for a major part of the message sending and time synchronization between federates in the federation. This is shown in Figure 1.

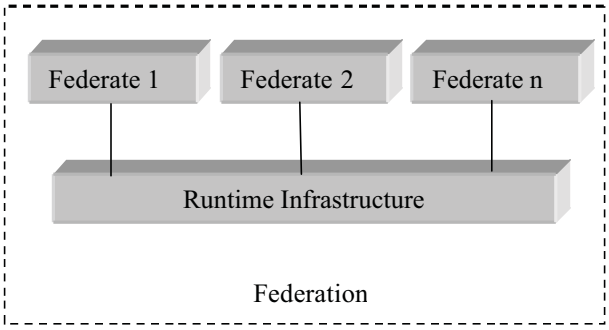

Figure 1. HLA Federation

The U.S. DoD states that HLA is based on the premise that no one simulator can satisfy all uses and users. An individual simulation or set of simulations developed for one purpose can be applied to another application under the HLA concept [2][3]. The HLA is a framework that supports modeling and simulation. "The HLA is the glue that allows you to combine computer simulations into a larger simulation."[4]. Although HLA has been created; it in itself is not a complete package and therefore cannot connect models together. The HLA is only a protocol that is enforced, as a standard on the developers of the RTI. There are rules that the HLA enforces for a design to become HLA compliant. These rules affect the design of the federate and federation [5].

4 The Modeling Process

The advantage of developing sub-models and then unifying them at runtime is that any number of people can work on developing a section of the overall model at any given time; allowing for more rapid development time. This does however require more planning to be performed before the modeling can commence.

Identify the problem
Determine how the model is to be split
Attributes that are required to be sent and received must be defined. (such attributes may never be requested, however if there is a chance that they may, then the sections model must be able to provide them).
Formulate the problem
Collect and process real system data
Formulate and develop a model – How entities flow through the system
Revise the Attributes that are to be sent and received
Validate the model - Compare the model's performance with known conditions of the real system.
Document model for future use – document the model in detail.
Select appropriate experimental design
Establish experimental conditions for runs
Perform simulation runs
Interpret and present results
Recommend further course of action

These steps are used for the creation of a unified simulation model and should be followed for successful creation of the model. There are extra steps that were added in to accommodate that unified model as 'normal' model creation has no need of working out how a model is to be split and the attributes required.

The main advantages of keeping a model all together is that the information can easily be obtained at the last minute, making models easier to change when they are nearer to completion. The advantage of a linked model is that any number of people can be working on a section of a model at any given time; allowing for more rapid development time. This does however require more planning to be done before the modeling can commence. The extra planning helps reduce unnecessary data collection.

5 Splitting a Model

Once the outline of the area to be modeled is defined, the next step is to divide this larger area in smaller sections. These smaller sections can then be modeled using different discrete event modeling packages and run even on different computers. The area to be modeled may be easy to split due to their obvious physical characteristics (automation islands) which can create; by default simulation islands. These simulation islands can be used to the advantage of the modelers and can save time when splitting models.

Sometimes, however, the boundaries for the models will not be so self evident. Such models that can be difficult to split are large networks of conveyor systems or models with a high number of resources (labour, AGV's or transporters) that move between separate work cells. An example of a difficult model to split is of two work cells connected with an AGV or a transporter. This can be seen in the image in Figure 2.

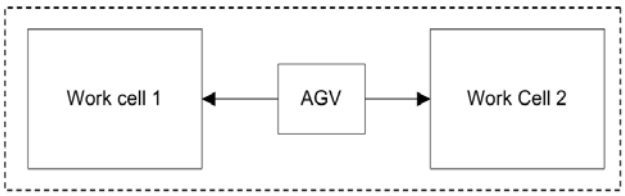

Figure 2. Standard way to model two work cells

This diagram shows the traditional single model method where everything is encapsulated within a single model (i.e. it would be modeled using a single modeling package). There are several other ways that this model could be split. The AGV could be modeled with one of the work cells and the second work cell could be modeled separately. This is shown in Figure 3.

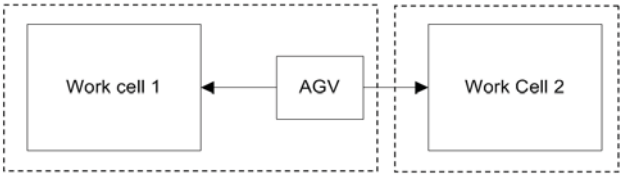

Figure 3. AGV in one model with work cell

Although this method may work it is crude and not very effective as the model that has the AGV has more control over the AGV by default. This is however not desired as both models may want equal control. If work cell 1 is sending a part to work cell 2 and work cell 2 has no say on when it receives the part this technique may be successfully applied.

The work cells can share the AGV, shown in Figure 4; however this technique does have its problems. Trying to get two different sections to have shared control over the AGV causes problems, eg. work cell 2 may send a request to the AGV in its model and work cell 1 may send a request to the AGV in its model. The problem lies in trying to get the AGV in each section to do the same thing. This technique is difficult to implement and adds difficulty for little or no benefit. It is also additional unnecessary work as the AGV must be modeled twice, i.e. more work for no real benefit.

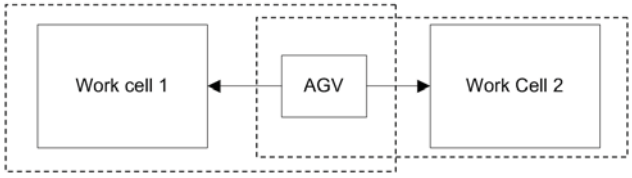

Figure 4. AGV modeled and shaded in both work cells

Another technique of sharing the AGV is to have half the AGV system modeled with work cell 1 and the other half modeled with work cell 2 as shown in Figure 5. While this technique may prove to be more accurate as to what is really happening. The problem occurs as before when each work cell sends a request to the AGV, it can only be in one model at a time. Once again this technique can increase difficulty for little or no benefit in return.

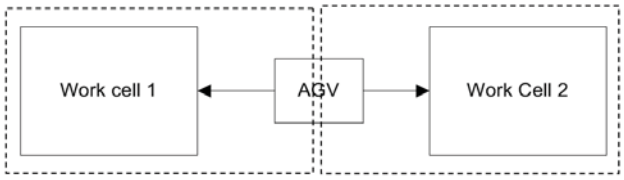

Figure 5. AGV shared in the model, split in half

Another more balanced procedure for solving a problem of the work cells and AGV is to have them all in their own separate model as shown below in Figure 6.

This technique makes three separate sections out of the model. It has work cell 1 as section 1, work cell 2 as section 2 and the AGV as section 3. This technique allows the most flexibility for modeling each part in a different simulation package if so desired. This also stops either of the work cells from having more control over the AGV due to its being modeled in the same section model as the work cell itself.

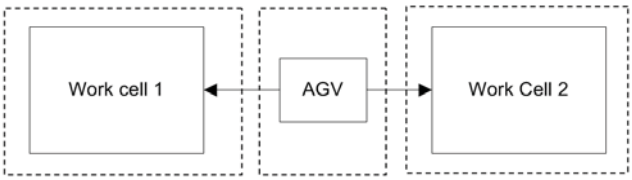

Figure 6. work cell 1, work cell 2 and AGV in different models

The decision of which way to divide the models depends on the characteristic of the system that is to be modeled. However, there is great difficulty when resources are shared or must cross into other simulation models as shown in Figures 4 and 5, which makes these designs unsuitable. The designs shown in Figure 3 and Figure 6 are a more suitable split as the resources are not shared;

however the design in Figure 3 still has the problem of one section having more control over the AGV. Because of these issues the method developed for unifying models requires that models be divided in such a way that resources do not cross the boundaries. Therefore, the information being passed among models primarily focuses on the parts (i.e. workpieces) that flow between models and status information of the resources.

6 Worked Example

6.1 Define Model Area

This section of the paper will describe and demonstrate the application of the previously discussed ideas to an example problem. As mentioned in the design procedure the first step that needs to be done is deciding the area that needs to be modeled. For this example, the following is to be modeled. The example that will be used is of an area that machines three separate parts that are then assembled by another machine in a separate location.

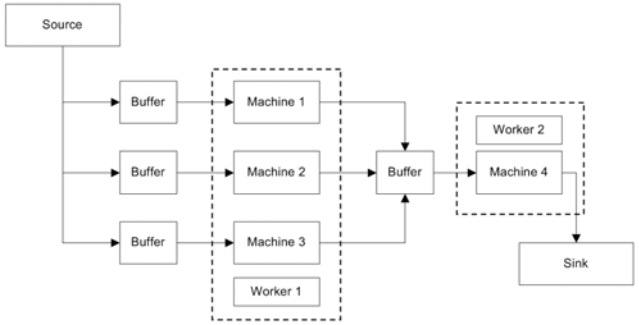

Figure 7. Layout of Area to be modeled

Figure 7 shows the layout of the area that is to be modeled. This area is designed to have three machines 1, 2 and 3 work on parts 1, 2 and 3 respectively. These parts are worked on for some period of time and then sent on to machine 4. Machines 1, 2 and 3 are operated by worker 1, who is responsible for all three machines. Machine 4 is operated by worker 2 and has the role of assembling the three parts together. Machine 4 requires a dedicated worker as the machine is an assembly operation. The assembled part is the final product that is sent to the sink (customers) finalizing the model.

6.2 Splitting the model

Next the system must be divided into smaller sections that will be modeled separately. The model will be split as can be seen in Figure 8. This split could have been done in a number of different ways, however, this seemed like the logical way as no resources cross this boundary. It could have however, been split so that each machine was in its own model, though this was decided against as this would create a lot of small models for little extra benefit. The section of the system with 3 machines is modeled using Quest and the other part is modeled using Arena.

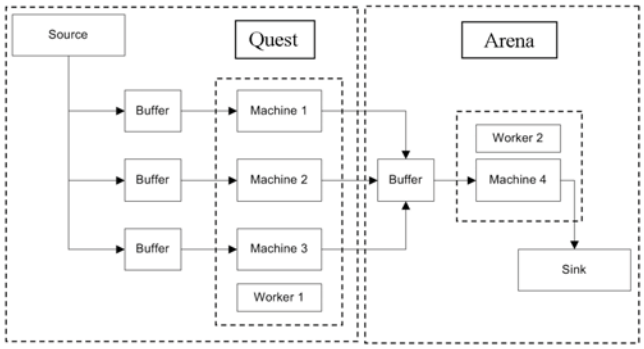

Figure 8. Arena that is to be modeled

6.3 Model Information

The next step is to determine what attributes, information and objects are needed to be sent from one model to another. The most obvious is the physical parts. As there are four parts in this model, (three that move from the Quest model to the Arena model and one part of the three assembled parts) then these four parts should be known about in both model parts.

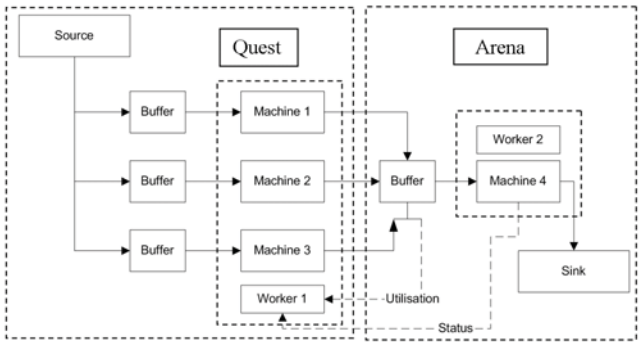

Figure 9. Information flow between Arena and Quest

Machines 1 ,2 and 3 may need to know the status of machine 4. Therefore machine 4 will need to make the status attribute available. When machine 4 receives a part from one of the machine in the Quest model; the Arena model sends the status of machine 4 and the number of parts back to the Quest model. The number of parts that are waiting in a buffer in the Arena model is needed as the decisions made by the worker in the Quest model depend on this information. Likewise, for the status of Machine 4,. Figure 10 shows the flow of information through the model via the broken arrows. The solid arrows show the flow of parts through the model.

6.4 Creating the Models

The Quest model (Figure 10) has been set up with the following parameters.

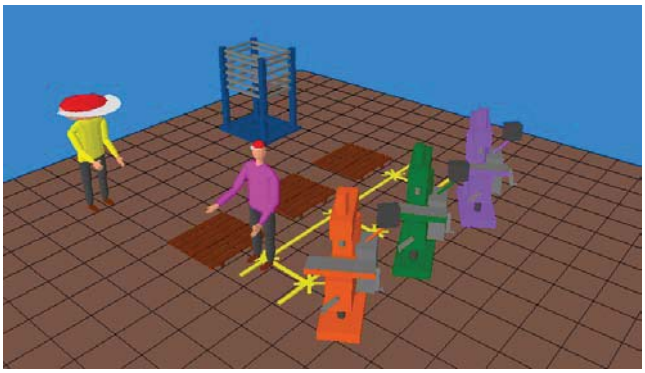

Figure 10. Quest model

machine 1's process time is for 50 seconds
machine 2's process time is for 100 seconds
machine 3's process time is for 150 seconds
parts arrive at source every 50 seconds
simulation is to run for 1000 seconds
request the status of the machine in the Arena model
request the number of parts (1,2 and 3) in the Arena model
use default worker logic (for worker and controller)
must know the different types of note. (status and number of parts) that will be sent to it via Arena.

The Arena model (Figure 11) must do the following.

wait until there is one of each part for the machine
Process these parts for 50 seconds to create a part
simulation is to run for 1000 seconds
have the number of parts available to be sent to the Quest model
have the machine status available to be sent to the Quest model
every time the model receives a part it is to send the status of machine 4
every time the model receives a part it is to send the number of parts it has (part 1,2 and 3).

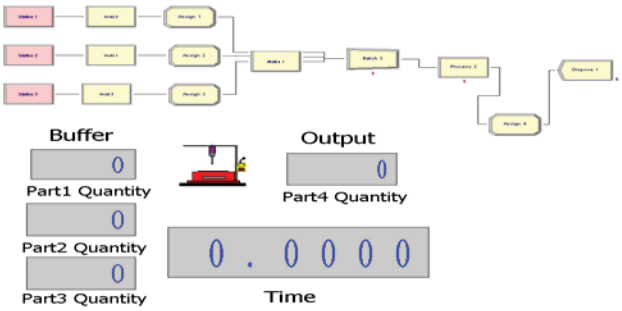

Figure 11. Arena model

6.5 Passing and Receiving Parts

Once the models are created using the linking methods that have been developed during this research, the simulation can be run. The majority of the message passing that occurs during this simulation is due to the passing of parts from one model to the next. Figure 12 shows the Quest model just as it is passing Part 1 to the Arena model.

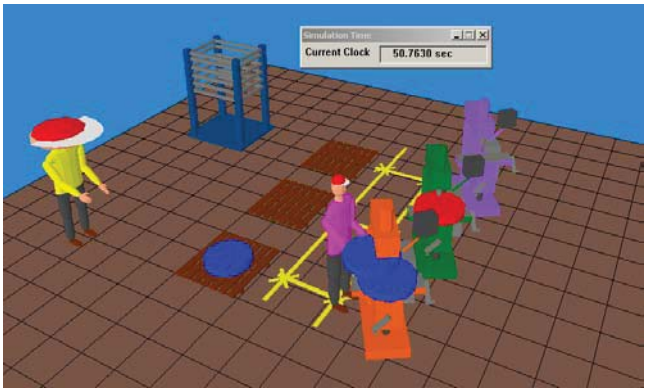

Figure 12. Quest model passing Part 1

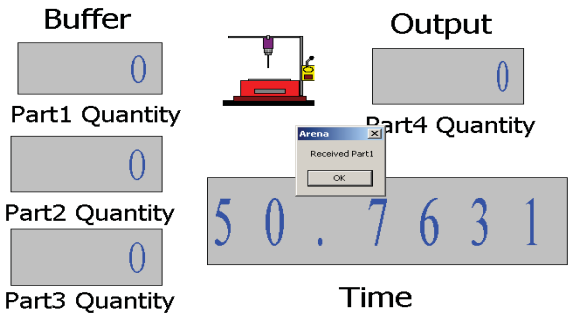

Figure 13. Arena model receiving Part 1

For the benefit of the worked example the Arena model has a message box appear to show that the model has received a part. The message also shows what part has been

received. Figure 13 shows the Arena model alerting that Part 1 is arriving in that model.

6.6 Assembling Part 4

Once Arena has received one of each part then Machine 4 can assemble these parts into Part 4. This operation is shown in Figure 14 as the machine in Arena shows the busy image. Figure 15 shows that there is one Part 4 in the out parts pile and that the first of the assembled parts has been created.

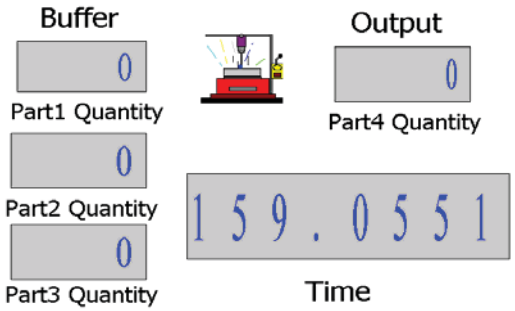

Figure 14. Arena working on Part 4

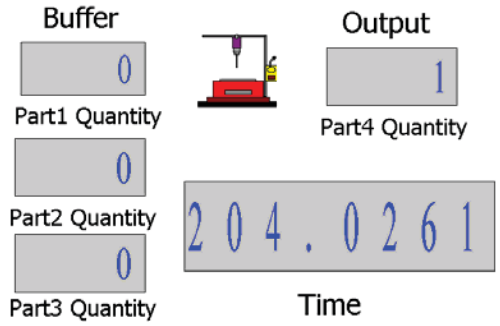

Figure 15. Arena finished Part 4

6.7 Passing Status Information

Figure 16 shows the display of the Quest model as it outputs the information that the Arena model has sent it. There is two types of information that is being sent to the Quest model. This information is the status of Machine 4 and the number of parts that are waiting in the buffer (not including the output parts or the parts that have been worked on). In Figure 16 it can be seen that there is information that has been passed from the Arena model as it receives a part, which is displayed in a Quest message box for demonstration purposes.

7 Conclusions

This paper demonstrates the creation and operation of a unified simulation model created by linking two sub-models at runtime. At the present stage this research allows for information to be obtained from the models allowing it to be passed to another model. The linking method using HLA has thus far been implemented and tested on several small models and found to work successfully. Further work is progressing with applications to larger models underway.

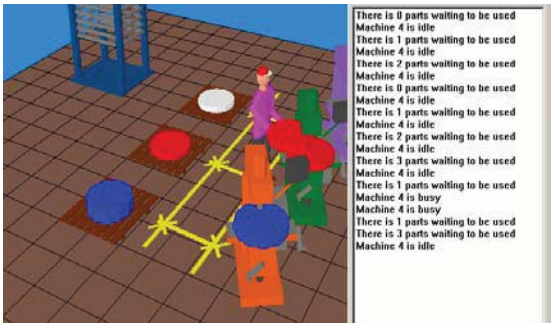

Figure 16. Quest receiving status and part number information

References

[1] Fujii, S. Kaihara, T. Mortia, H. 'A distributed virtual factory in agile manufacturing enviroment', *International Journal of Production Research*, 2000, volume 38, no 17, 4113-4128

[2] United States Defence Modeling and Simulation Office https://sdc.dmso.mil/

[3] Dahmann, J., Fujimoto, R., Fujimoto, R., 'The Department of Defence High Level Architecture', *In Proceedings of the 1997 winter simulation conference,* ed., Andradóttir, S., Healy, K. J., Withers, D. H., Nelson, B. L., Atlanta, GA

[4] Kuhl, F., Weatherly, R. and Dahmann, J. S. 'Creating Computer simulation systems: An introduction to the High Level Architecture', 1999, Prentice Hall PTR.

[5] IEEE standard 1516-2000, 'IEEE Standard for Modeling and Simulation (M7S) High Level Architecture (HLA) – Framework and Rules, Approved 21 September 2000

[6] Charles, McLean., Frank, Riddick, 'The IMS mission architecture for distributed manufacturing simulation', in *Proceedings of the 2000 Winter Simulation Conference*, ed., Joines, J., Barton, R., Kang, K., Fishwick, P. A., Orlando, Florida, USA

[7] Kuhl, F., Dahmann, J. S., Weatherly, Richard., 'Standards for Simulation: As Simple As Possible But Not Simpler The High Level Architecture For Simulation', *Simulation* 1998, 71:6, 7 378-387.

[8] Hibino, H., Fukuda, Y., 'A Study on Support System for Distributed Simulation Systems of Manufacturing Systems Using HLA', *The 5th International Conference on Design of Information Infrastructure Systems for Manufacturing 2002*

[9] Hibino, H., Fukuda, Y., Yura, Y., Mitsuyuki, K., Kaneda, K., 'Manufacturing Adapter of Distributed Simulation Systems Using HLA', *In Proceedings of the 2002 winter simulation conference*, ed., Snowdon, J. L., Charnes, J. M., Yücesan, E., Chen, CH, San Diego, CA

[10] Hibino, H., Fukuda, Y., Yura, Y., Nakano, M., Fujii, S., 'A Study on Distributed Simulation Systems to Evaluate Manufacturing Systems Using HLA', *2002 Japan - USA Symposium on Flexible Automation*

[11] Price, D., Walsh, S., Nahavandi, S., 'Unifying Manufacturing Simulation Models Using HLA', *In Proceedings of 2nd IEEE International Conference on Industrial Informatics,* ed., Schoop, R., Colombo, A., Bernhardt, R., Schreck, G., Berlin Germany, pp.190 -195 June 2004.