Proceedings of the 2002 Winter Simulation Conference
E. Yücesan, C.-H. Chen, J. L. Snowdon, and J. M. Charnes, eds.

DEVELOPMENT OF DISTRIBUTED SIMULATION MODEL FOR THE TRANSPORTER ENTITY IN A SUPPLY CHAIN PROCESS

Richard J. Linn
Chin-Sheng Chen
Jorge A. Lozan

Department of Industrial and Systems Engineering
Florida International University
Miami, FL 33199, U.S.A.

ABSTRACT

Transporter is a critical part of Supply Chain integration. An international transporter process involves multiple ground pickup and delivery operations, package sorting and palletizing, airport operations and air transport. This paper describes a successful two-machine implementation of a distributed simulation model for an international transportation system in a supply chain network operation using Run Time Infrastructure of High Level Architecture software developed by the Defense Modeling and Simulation Office, the Distributed Manufacturing Simulation Adapter developed by the National Institute of Standards and Technology, and ARENA simulation tool. By incorporating the capabilities provided by these tools, it was successful to establish the information flow exchange between the machines where one machine houses transporter while the other has suppliers, customers, and distribution centers located in different parts of world. This research tool attempts to facilitate the development of distributed simulations so they can be used to analyze and solve manufacturing related problems.

1 INTRODUCTION

An effective Supply Chain Management (SCM) means integration of all functions such as sourcing, procurement, production scheduling, warehousing, and transportation, so that the material and information flow continuously from firms to firms. The SCM integration must be done through the information. The idea is to have information trail that follows the product's physical trail so that the planning, tracking, and lead-time estimation, etc. can be done on real time data. Any party who has the need to know where the product is should be able to access the data.

A firm involved in a supply chain will always have an up-stream supplier network and downstream distribution network. For the distribution network of manufacturer, different entities are involved: customers (buyers), distributors/retailers, and transporters. Manufacturer may ship through a transporter to distributors first. Then retailer will receive his supply from the distributor. Multi-mode transports, such as land/sea or land/air, are often employed between the two entities. This is particularly true if international supply chain is involved. Multi-mode transport involves multiple operations such as freight forwarders, trucking companies, terminal operators and carriers (ocean-liner, airlines). An effective integration of all these operators requires a carefully planned strategy. To model such SCM operation, the distributed simulation modeling that connects through Internet multiple simulation models residing on different machines at a distance has been proposed.

This project is to develop a transporter simulation model that can be easily integrated with other simulators through Internet using the readily developed Distributed Manufacturing System Adaptor (DMS Adaptor) from National Institute of Standards and Technology. The objective of this report is to describe the development of a distributed simulation model for a transportation system that includes multiple transporters. DMS Adapter is imbedded to establish information flow exchange between simulation models.

2 SUPPLY CHAIN PROCESS

The modeling of the supply chain system simulates a SCM process of four international locations. These are the operations located in Miami, Lima, Honduras, and Venezuela. The general process are described here.

2.1 Local Operation

1. In any of the four locations, customers place purchase orders to their local Distribution Center (DC). The order information includes:
   a. Type of order,

b. Order amount,
c. Due date,
d. Customer's preferred supplier location (if necessary),
e. Customer's current city, and
f. Customer's zone of residence.

2. DC receives the order and proceeds to check if inventory is available. If there is, DC prepares shipment to fill the order from its inventory and creates a shipping order to the transporter whose trucks will come to pick up. Otherwise, the purchase order is placed on hold until inventory replenishment arrives again.
3. DC inventory's internal system verifies whether a reorder point is reached. At every purchase order, the internal system decides whether or not to trigger the automatic reorder process to their correspondent supplier.
4. All shipments are held until 8:00 pm until transporter's pickup takes place.
5. During the pickup, the items are checked, packing list is attached, shipments are loaded and pickup is confirmed with the shipper. The package is now officially in the transportation system. The transportation method will be local because shipment is going to local customers.
6. Through the local transportation (trucking) system, packages are delivered to customers. Customers are partitioned into zones according to their local addresses. A truck will handle both pickups and deliveries and is assigned to a zone only. Currently, each city is divided into three major zones.

2.2 International Operation

1. When DC reaches its re-order point, a purchase order is placed to its supplier, which may be located in a different country. The supplier receives the purchase order from the DC and proceeds to verify its inventory. If inventory is available, Supplier fills the order from its inventory by creating the shipment for the purchase order and a shipping order to the transporter. Otherwise, the purchase order is placed on hold until the replenishment arrives.
2. Supplier inventory's internal system verifies whether a reorder point has been reached. Whenever a purchase order is received, the internal system decides whether or not to trigger the production for inventory replenishment.
3. Shipments are held up until 8:00pm every night until transporter's pickup takes place.
4. At pickup, shipments are checked whether or not they are local or international. The supplier location may not necessarily have the same location as the distribution center. For instance, a customer that lives in Lima wishes to buy a pair of snickers manufactured in Miami. However, DC's inventory located in Lima does not have that product in stock. Thus, they will contact the supplier in Miami to complete the customer order. Hence, the transportation system checks if the shipments will be transported to a local DC or sent overseas. Upon exiting the transportation system, packages are delivered to the distribution centers accordingly.

3 TRANSPORTER OPERATION

International logistics involves local SCM operations in different countries and international carriers connecting them. Figure 1 presents the relationship between the local SCM operations and the transportation system. Our current study deals with the following scenario:

1. A customer sends purchase order to a DC. Upon receipt of the purchase order, the local DC prepares the cargo/package shipment and awaits pickup service to collect them.
2. The packages collected from a DC are brought to transporter's regional or local sorting center. The packages are processed, scanned, sorted, bagged, and palletized according to their destinations.
3. If the packages are designated for local address, the processed packages are delivered through transporter's local delivery to the customer's address.
4. Otherwise, the packages are transported to the airport terminal.
5. At the terminal, the shipments are to clear customs and placed onto dollies for the ramp service to load them into the proper carrier.
6. Once reached their destination points, the containers are unloaded at the terminal, and broken down into individual shipments for further inspection by customs.
7. Once cleared, they will be transported by vans to its regional sorting center for later delivery to distribution center.
8. From the distribution center, packages follow the exact same process explained in steps 1 and 3.

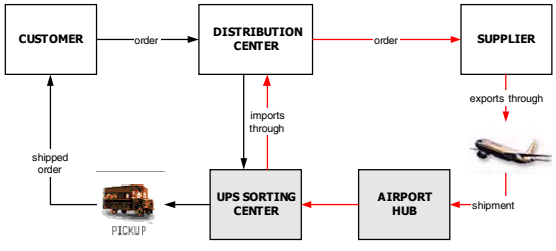

Figure 1: SCM and Transporter system flow

3.1 Transporter's Ground Operation and Carrier Network Development

The entire transportation system includes an air career network (Figure 2) connecting a ground operation of each airport in different regions. Carriers transport cargoes and small packages from airport to airport. Connected to each airport is a series of ground transporter operations. Each ground operation is divided into import and export operations. Everyday, according to the packages to be delivered in each zone and pickup locations to visit, trucks will be dispatched from sorting center to deliver and pick up packages. At the pickup locations, export shipment enter the transportation system. At the delivery location, import shipments leave the transportation system. Trucks bring export shipments back to the sorting center where the collected packages are sorted according to their destination and flight numbers. For each operation, it is necessary to determine the information transferred between operations, the process involved in converting the information from input to output within each operation, and the capacity at each operation (Linn and Chen, 2001).

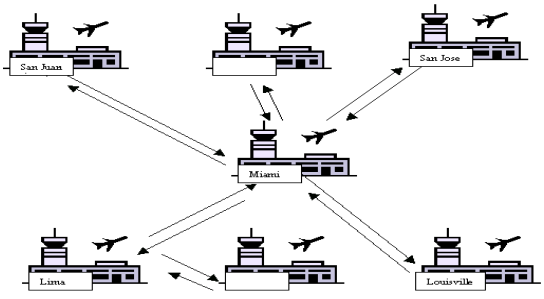

Figure 2: Air carrier network connecting cities/airports

4 HLA/RTI Software and the DMS Adapter

The Run Time Infrastructure (RTI) is based on the High Level Architecture (HLA) developed by the Defense Modeling and Simulation Office (DMSO) to provide a consistent approach for integrating distributed, defense simulations. An HLA-based distributed simulation is called a federation. Each simulator or system that is integrated the federation is called a federate. One common data definition is created for domain data that is shared across the entire federation. It is called the federation object model (FOM). Each federate has a simulation object that defines the elements of the FOM that it implements.

There is, however, no interoperability across RTI implementations (McLean and Riddick, 2000). Hence, a distributed simulation running on different computer systems across a network must use the same RTI software as an integration infrastructure. The Distributed Manufacturing Simulation (DMS) Adapter is developed as part of the IMS Mission Project. The purpose of the adapter is to facilitate the development of distributed simulations for analyzing and solving manufacturing related problems (Riddick, 2001). The DMS Adapter is incorporated into each DMS federate. The adapter will handle the transmission, receipt and internal updates to all the FOM objects used by a federate. The DMS adapter provides a simplified time management interface, automatic storage for local object instances, management of lists of remote object instances of interest, and management and logging for interactions of interest. The DMS Adapter is meant to provide mechanisms for distributed simulation similar to those that are provided by the HLA RTI, but with a level of complexity that is manageable by the development resources available in the manufacturing community.

Many objects in the FOM may reference documents containing more detailed information that are stored in a file system, PDM system, or database. The Extensible Markup Language (XML) can be used to define new document types (Goldfarb and Prescod, 2000). XML allows for the definition data that has semantic information in addition to the data values. XML data-type-definitions (DTD) may be used to define new document formats. Even without the DTD, XML files are often both human and machine readable because of the semantic information that is included. Moreover, there are potentially many document types that will be stored as distributed manufacturing simulation data. Examples of these include many kinds of CAD files, image files, and executables. However, many manufacturing documents do not have standardized format. Examples of these are schedules, bills of material, and process plans. XML presents itself as a mechanism to allow the definition of these extensible formats without affecting the DMS architecture or interfaces.

5 Model Development

The distributed simulation model development was divided into 6 steps:

1. Study SCM process of transporting packages and cargo in UPS International operation.
2. Develop generalized simulation model of the transporter operation.
3. Develop detailed information model of the transporter's ground operation and carrier network.
4. Study DMS Adapter and XML functionality.
5. Place the ground operation and carrier model in one computer, suppliers and distributors and customers (buyers) into second computer. Embed DMS Adapter in each model to develop Internet based multi-computer distributed simulation model.
6. Expand the information content of the simulation model according to the information model.

Steps 1 and 2 are explained in the previous sections. Thus, the following will be the description of the transporter system and the DMS Adapter.

5.1 Transporter's Ground Operation and Carrier Network

The simulation model developed after studying the process of transporting packages and cargo in UPS International operation follows this process flow.

Scenario #1: A supplier in this country ships orders to distribution center in another country

1. Export process begins when UPS trucks bring orders to local sorting center. The information processed here includes a package number, package type, value, weight, number of pieces, pickup time, and flight number.
2. Packages are then key-entered into the system, inspected, scanned, labeled, bagged, sealed, and placed in containers according to their destinations.
3. At 9:30p, Vans will transport the containers to the airport terminal.
4. At 10:10p, Vans arrive at the airport. Packages are unloaded from the vans and palletized based on the containers' destinations.
5. At 10:55p, packages are weighted and they will remain stacked until their flight departure times.
6. In this experiment, each of the four cities possesses a pre-determined schedule. For instance, flights from Lima will depart at 00:36a, flights from Miami at 01:36a, flights from Honduras at 02:36a, and flights from Venezuela at 03:36a. This is normal since flight, or service, schedule is often set for a long period of time.
7. During early morning, airplane carriers will reach their destinations in the various cities in the current carrier network. Export process ends.
8. Import process begins: At 7:00a, packages arrived from the airplanes are loaded onto pallets by ground operations and transported to Customs.
9. Packages are scanned and checked by Customs.
10. Upon clearing, packages will be sorted by types and values. For instance, packages whose monetary value exceeds $100 will require special attention.
11. At 9:00a, Vans pick up the released shipments from Customs and transport them to local sorting center (Hub).
12. Packages delivered to the local Hub are scanned and checked for problems such as repacking, bad addresses, and heavy traffic delivery or customer service issues.
13. At 10:00a, UPS truck picks shipments up and delivers them to local distribution center. Import process ends.

Scenario #2: A supplier in this country ships order to distribution center in the same country.

Scenario #3: A distribution center ships order to customer (it was already stated that DCs and Customers are in the same country).

1. UPS trucks deliver shipments to local Hub at approximately 8:30p.
2. Packages delivered to the local Hub are scanned and checked for problems such as repacking, bad addresses, and heavy traffic delivery or customer service issues.
3. If shipments are to be delivered to customers, they are palletized according to customer's zone of residence.
4. At 10:00a, UPS trucks picks shipments up and start delivering them to local distribution center and/or customers

5.2 Install the DMS Adapter

In order to enable the transfer of information flow between the simulation models, it is necessary to follow these instructions:

1. Download and Install the RTI software documentation.
   This version of the DMS Adapter is built to work with version RTI1.3NG-V3.2. To download, sign up with the DOD at `http://sdc.dmso.mil`. The DMS Adapter has been tested with WinNT 4.0 but it should work with Win98 and Win2000.
2. Set up environment variables for RTI_HOME, RTI_BUILD_TYPE, and RTI_RID_FILE in the System properties of the Control Panel.
   RTI_HOME corresponds to the path where the RTI executable files are located. For example: "C:\Program Files\ \DMSO\RTI1.3NG-V3.2"
   RTI_BUILD_TYPE corresponds to version name of the RTI software. For example: "Win2000-VC6"
   RTI_RID_FILE corresponds to the path of the RID file. For example: "C:\Distribution\BIN\RTI.rid"
3. Add the following string (without the quotes) to the end of the Path environment variable: ";%RTI_HOME%\%RTI_BUILD_TYPE%\bin;"
   NOTE: Test the RTI by opening the command prompt window and typing "rtiexec". If you get messages saying that there are missing DLLs, extract the file bin\MSdlls.zip and copy the DLLs to the winnt\system32 directory.
4. Extract the files from the zip files containing the DMS Adapter.
5. Run RegisterAdapter.bat in the bin directory to register DMSAdapter.dll. Run RegDebugLog.bat

in the bin\DebugLog directory to register DebugLog.exe.
NOTE: If the DMSAdapter will not register, unzip the bin\MSdlls.zip and copy the DLLs to the bin directory.

6. Customize the RTI address in the RID file and RTIexec.bat
   NOTE: Follow these steps if you attempt to run a federation on the same computer. Otherwise, refer to Running RTIexec in a LAN environment.
   In the RTI.rid file there is an address setup for the MulticastDiscoveryEndpoint. In the RTIexec.bat file in the bin directory, there is a command line parameter called "-multicastDiscoveryEndpoint" which must have the same address. The format of the address is:
   224.aa.bb.cc:ddddd, where aa, bb, and cc represent integers between 0 and 99. ddddd represents a port number.
   NOTE: an RTI process must be started before a federation can be created and can be only one RTI process on a network. If more than one person attempts to start a federation on the same network, change the address in the RID file and RTIexec.bat so that each person can have their own RTI.
7. Make sure the FED file (DMSAdapter.fed) is in the same directory as the RID file.
8. Start the customized version of RTIexec.bat before attempting to start any federations.

5.2.1 Running RTIexec.bat in a LAN/ WAN/Internet Environment

This approach can be used if all of the federates will be running on the same LAN, over a WAN or the Internet. It uses the "endpoint" parameter of the RTIexec and the RTIExecutiveEndpoint data in the RID file.

1. First, decide which machine to run the RTIexec on. Determine its real IP address and its complete hostname. Next, make up a port number. 12345 should work fine. The RTIexec will only need to be started on one of the computers. For our experiments, the modified RID file had a new line that looks like this:
   (RtiExecutiveEndpoint 131.94.167.13:12345)
   In the RTIexec.bat, change the endpoint parameter to:Rtiexec.exe –endpoint hunter.eng.fiu.edu: 12345
2. Make sure that all every federates will use the same RID file containing the new changes. That is, the RTIExecutiveEndpoint property must be set in each RID file that is used.

5.2.2 Running Simulations with the Federation Manager

The Federation Manager provides the capability to specify how many simulations must have reached the "Ready to Run" state before any are allowed to transition to the "Running" state. This basic capability is necessary for simulations to synchronize so that they all start running at the same time value, usually time zero. To use this Federation Manager with a federation, two steps must be done:

1. Start FedMan before any simulations have attempted to initialize. Execute FedMan at the command prompt and pass the number of federates that will be in the federation. For instance, if two federates will be part of the federation, use the following command:C:\Distribution\bin> fedman 2
2. Set the Adapter property "WithManager" to "Enable" on each adapter or federate before calling the Initialize method.
   If your reference to the Adapter is called "theAdapter", setting the proper value can be done with a VB statement like the following:
   rc = theAdapter.SetProperty("WithManager", "Enable")
   This must be done for all adapters in the federation.

5.3 Arena Model Integration using the DMS Adapter

In order for the DMS Adapter to work correctly within a simulation model developed in Arena, it is necessary to include a series of blocks that will enable the Adapter logic. Furthermore, Visual Basic statements are also needed to reference an instance object to the adapter. The following guidelines should provide a good start to integrating these two concepts.

5.3.1 Create the Adapter Logic in Your Model

As shown in Figure 3, start with a Create block, two VBA blocks, and a Delay block. The Create block will just create one single entity used for coordination. The first VBA block will be used for time advancement. The second VBA block will read and process the messages. The Delay block causes the simulation to delay until time needs to be advanced again. This delay will allow the model to go from 0 to 1 second (assuming the DMSAdapter's property SimulationStepSize has been set to 1000).

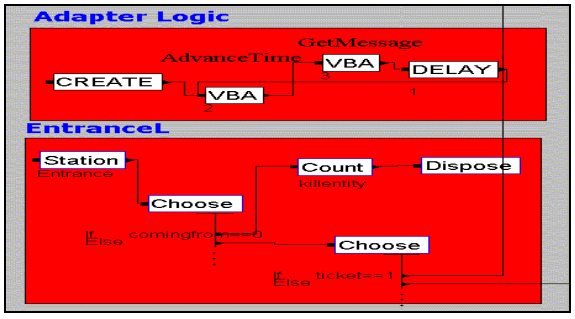

Figure 3: DMSAdapter Logic in SCM model

5.3.2 Create the VB Statements in the Visual Basic Editor

Depending on the VBA Cookie that Arena assigns to each VBA block, the VB statements will be as follows:

a. For time advancement:

```
Private Sub VBA_Block_2_Fire()
' Advance Simulation
 rc = theAdapter.AdvanceSimulation()
   'wait for the end of communication
   While theAdapter.SimulationAdvanceCompleted() = False
   Wend
 End Sub
```

b. For message reading:

```
Private Sub VBA_Block_3_Fire()
'check the messages one by one
  While theAdapter.AllMessagesReceived() = False
'receive the message from the adapter
   theStr = theAdapter.GetNextMessage()
   Call ProcessMessages(theStr)
   Call  gsiman.EntitySendToBlockLabel(gsiman.ActiveEntity,
nodelay, "EntranceL")
   Call  gsiman.EntitySendToBlockLabel(gsiman.EntityCreate,
nodelay, "GetMessageL")
  Wend
End Sub
```

For our experiments, the calls of sub-procedures were inserted into this subroutine.

c. Calling SIMAN procedures:

At this point, it is important to notice the two SIMAN procedures and the logic behind them.

The first procedure: Call gsiman.EntitySendToBlock Label(gsiman.ActiveEntity, nodelay, "EntranceL") is called right after processing the DMSAdapter messages. It creates a new entity and sends it to the block labeled "EntraneceL" with no delay of time. Variable 'nodelay' was defined previously and assigned the value 0.

The second procedure: Call gsiman.EntitySendTo BlockLabel(gsiman.EntityCreate, nodelay, "GetMessageL") sends the current entity to the same VBA_3 block in order to keep reading messages from the rest of the federation. Coincidently, the VBA_3 block was labeled "GetMessageL". So, after all the messages have been read, the current entity will continue the DMSAdapter logic.

Now, refer to Figure 3. Station block 'Entrance' is labeled "EntranceL" wherein the new created entity will be sent. The reason why there is a Choose block is that during the simulation run, the DMSAdapter may sent messages that are not relevant to the study of this project. For instance, every time a federate joins the federation, a notification type message is sent automatically. If my simulation model receives that message and processes it, a new entity will be created with no attributes at all since their XML nodes did not include any data concerning the SCM process. Thus, by setting a condition 'If comingfrom>0' the SCM model will know for certain that an entity that checks true for that conditions has an attribute called 'comingfrom'.

Now, refer to Figure 4. This logic corresponds to the transporter model. It follows the exact same logic of the SCM model. It disposes irrelevant entities before they enter the system. However, it is necessary to distinguish the entities that come from the Supplier and Distribution Center nodes and redirect them to their proper station blocks. Some may go to the Export process wherein they are sent to a local sorting center and then transported to the airport terminal. Some may go to a regional hub center, sorted and placed for later delivery by the UPS trucks. The attribute 'ticket' takes care of that issue. Ticket value 1 means that the entities pertain to the distribution center and must be routed to the local hub. Ticket value 2 means that the entities come from the supplier node. Special attention must be given at this point now because it is necessary to differentiate whether the supplier and distribution center are located in the same country. The condition 'If comingfrom==mysupplier' determines whether they both are in the same city. If so, those entities are routed to the local hub. Otherwise, they must follow the Export process and be transported through the carrier network to their destination point.

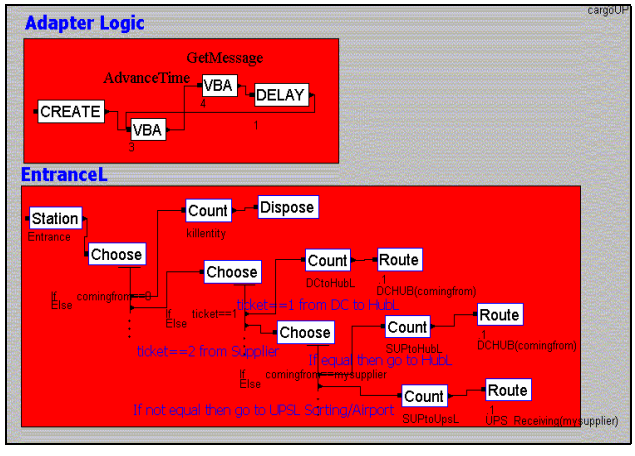

Figure 4: DMS Adapter logic in Transporter model

d. For message processing:

```
Public Sub ProcessMessages(theMsg As String)
Dim myDoc As New MSXML.DOMDocument
Dim anode
Dim elemlist, i, myticket, mycomingfrom, myduedate, mysup-
plier, myzone, myorderamount, myordertype, mytoa
' Load XML
myDoc.loadXML (theMsg)
' Retrieve attributes from messages
Set elemlist = my-
Doc.getElementsByTagName("ComingFrom")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  mycomingfrom = anode.Text
  Next
Set elemlist = myDoc.getElementsByTagName("DueDate")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  myduedate = anode.Text
  Next
Set elemlist = myDoc.getElementsByTagName("Supplier")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  mysupplier = anode.Text
  Next
Set elemlist = myDoc.getElementsByTagName("Zone")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  myzone = anode.Text
  Next
Set elemlist = my-
Doc.getElementsByTagName("OrderAmount")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  myorderamount = anode.Text
  Next
Set elemlist = myDoc.getElementsByTagName("OrderType")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  myordertype = anode.Text
  Next
Set elemlist = myDoc.getElementsByTagName("Ticket")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  myticket = anode.Text
  Next
Set elemlist = myDoc.getElementsByTagName("TOA")
  For i = 0 To (elemlist.length - 1)
  Set anode = elemlist.Item(i)
  mytoa = anode.Text
  Next
' Assigning attributes to entity obtained from XML
gsiman.EntityAttribute(gsiman.ActiveEntity, gcomingfrom) =
mycomingfrom
gsiman.EntityAttribute(gsiman.ActiveEntity, gduedate) = my-
duedate
gsiman.EntityAttribute(gsiman.ActiveEntity, gmysupplier) =
mysupplier
gsiman.EntityAttribute(gsiman.ActiveEntity, gmyzone) = my-
zone
gsiman.EntityAttribute(gsiman.ActiveEntity, gorderamount) =
myorderamount
gsiman.EntityAttribute(gsiman.ActiveEntity, gordertype) =
myordertype
gsiman.EntityAttribute(gsiman.ActiveEntity, gtoa) = mytoa
gsiman.EntityAttribute(gsiman.ActiveEntity, gticket) =
myticket
End Sub
```

So far, this explanation ensures effective one-way communication. However, a two-way communication is imperative for this type of simulation study. In order to enforce this, it is only necessary to insert a VBA block right before the entity is disposed from the system. For instance, in the transporter model a VBA block was inserted right after the UPS trucks deliver the shipments to their corresponding end nodes and before the entity reaches the Dispose block. The following VBA statements convert all the relevant attributes that the entity owns into XML data and concatenate a XML message that it is sent to the other federates. Keep in mind that additional SIMAN methods are included and used in previous sub-procedures of the model logic. Moreover, all the variables used must be put in the declarations section and all the object instances referencing the MSXML and DMSAdapter library must be created.

e. For message sending:

```
Private Sub VBA_Block_4_Fire()
Dim miticket, micomingfrom, miduedate, mimysupplier, mi-
myzone, miorderamount, miordertype, mitoa As Double
Dim theMsgType
Dim theMsgData
micomingfrom = gsiman.EntityAttribute(gsiman.ActiveEntity,
gcomingfrom)
miduedate = gsiman.EntityAttribute(gsiman.ActiveEntity,
gduedate)
mimysupplier = gsiman.EntityAttribute(gsiman.ActiveEntity,
gmysupplier)
mimyzone = gsiman.EntityAttribute(gsiman.ActiveEntity,
gmyzone)
miorderamount = gsiman.EntityAttribute(gsiman.ActiveEntity,
gorderamount)
miordertype = gsiman.EntityAttribute(gsiman.ActiveEntity,
gordertype)
mitoa = gsiman.EntityAttribute(gsiman.ActiveEntity, gtoa)
miticket = gsiman.EntityAttribute(gsiman.ActiveEntity,
gticket)
theMsgType = "ProcessOrder_SUPPLIER"
theMsgData = "<ComingFrom>" + Str(micomingfrom) +
"</ComingFrom><DueDate>" _
+ Str(miduedate) + "</DueDate><Supplier>" +
Str(mimysupplier) + _
"</Supplier><Zone>" + Str(mimyzone) +
"</Zone><OrderAmount>" _
+ Str(miorderamount) + "</OrderAmount><OrderType>" +
Str(miordertype) + _
"</OrderType><TOA>" + Str(mitoa) + "</TOA><Ticket>" +
Str(miticket) + "</Ticket>"
rc = theAdapter.SendMessage(theMsgType, theMsgData)
End Sub
```

6 CONCLUSIONS

To model an integrated SCM operation, he distributed simulation modeling that connects through Internet multiple simulation models residing on different machines at a distance has been proposed. The first phase of the research emphasized primarily with the proper integration of the Arena simulation models using the DMS Adapter. A UPS International package delivery scenario was chosen to test

the functionality and behavior of the DMS Adapter. Two computers were used to test this tool. Thus, one computer simulated the behavior of the supply chain network and the second computer dealt with the transportation system. Based on the initial specifications of the supply chain network, the DMS Adapter was used effectively to communicate to the second simulation model in the federation network. Upon receipt of these messages by the second machine, several operations occurred according to the fixed schedule of the transportation system. Basically, these messages contained specific instructions that triggered the initiation of the transportation model such as order types, order amounts, due dates, city and airport destinations, etc. It is important to mention that the use of the XML language in the messages facilitated the creation and format of those messages and thanks to Microsoft's XML Object Model compatible with Arena, it was possible to access and retrieve the data. The two-way communication finalized when a confirmation message was sent to the first computer notifying that the shipments were ready to be delivered. This implementation has successfully utilized the DMS Adaptor's capability of providing distributed simulation integration through the Internet.

7 FUTURE DIRECTIONS

At this point, the connectivity of the distributed simulation can be achieved through the DMS Adaptor. The scope of the second stage of the research agenda will be as follows:

1. Expand the supply chain models into multiple machines so that The transporter model will actually interacting with multiple supply chain entity models.
2. Create basic simulation blocks and elements that can be used to build transporter instances in a supply chain for different applications.
3. Explore the configuration strategy over the Internet to allow the use of distributed simulation as a configuration tool for logistics service integration, or more recently fourth party logistics.

ACKNOWLEDGMENT

The work is supported by the Intelligent Manufacturing Systems Group, National Institute of Standards and Technology, and United Parcel Service, Miami, Caribbean and Latin America operation.

REFERENCES

Goldfrab & Prescod, 2000, *The XML Handbook*, Prentice Hall: Upper Saddle River, NJ

Linn, Richard and Chen, Chin-Sheng, 2001, *Development of Distributed Simulation Model for a Transport System in a Supply Chain Process*, a report submitted to the National Institute of Standards and Technology, Department of Industrial and Systems Engineering, Florida International University.

McLean, Charles and Riddick, Frank, 2000, "The IMS Mission Architecture For Distributed Manufacturing Simulation," *Proceedings of the Winter Simulation Conference 2000*, Orlando, FL.

Microsoft XML 3.0 – XML Tutorial and XML Reference http://msdn.microsoft.com/library/en-us/xmlsdk30/htm, accessed on July 15, 2001.

Riddick, Frank, 2001, *The Distributed Manufacturing Simulation Adapter Reference Guide*, National Institute of Standards and Technology.

AUTHOR BIOGRAPHIES

RICHARD LINN is an associate professor of Industrial and Systems Engineering at Florida International University. He received his PhD from Penn State in 1987. His areas of interest are logistics, transportation management and production planning.

CHIN-SHING CHEN is a professor of Industrial and Systems Engineering at the Florida International University. He received his Ph.D. from Virginia Tech in 1985. His areas of interest are Concurrent Engineering, CAD/CAM, Automation, and Applied Operations Research.

JORGE LOZAN is an MS student of Industrial and Systems Department at Florida International University. His focus is on the simulation and information systems.