International Journal of Industrial Engineering, 11(2), 151-160, 2004.

DESIGN AND DEVELOPMENT OF A PROTOTYPE DISTRIBUTED SIMULATION FOR EVALUATION OF SUPPLY CHAINS

Jayendran Venkateswaran, Young-Jun Son

Department of Systems and Industrial Engineering
University of Arizona
1127 E. North Campus Drive
Engineering Building #20, Room 111
Tucson, AZ 85721
Email: Young-Jun Son, son@sie.arizona.edu

In this paper, a prototype distributed simulation system is proposed to evaluate viability of a simulated supply chain. Members of a supply chain, activity definitions for each member, and information and material flow among members are discussed. IDEF∅ functional modeling tool has been used to model the functions of the system and the relationships among the functions. The interaction between the members of the system is then illustrated using time-sequence diagrams, and the behavior of each member is represented using the deterministic finite state automata. These formal models have formed the basis for the development of the distributed simulation system. Reusable simulation models for each of the members of the system have been developed using commercial simulation tools such as Arena™ and ProModel™. The High Level Architecture (HLA) Run Time Infrastructure (RTI) has been used to provide an interface to create the distributed simulation system. Preliminary performance tests have been conducted to evaluate the suitability of the proposed system in the Internet environment.

Significance: This paper addresses the application of distributed simulation technology to evaluation of potential supply chains. The use of distributed simulation technology allows each potential partner to hide any proprietary information in the implementation of the individual simulation, but still provide enough information to evaluate the supply chain as a whole.

Keywords: Simulation, Distributed simulation, IDEF, Supply chain.

(*Received 20 September 2001; Accepted in revised form 31 July 2002*)

1. INTRODUCTION

Today's manufacturing industries face the challenge of responding more rapidly and efficiently to changing markets driven by customized products. The agile manufacturing paradigm has been proposed to resolve this problem. Agile manufacturing is a technology for achieving flexibility and rapid responsiveness to changing market and customer needs. Agile manufacturing enables a firm to quickly respond to customers' requirements and then design, prototype, manufacture, test and deliver a high-quality product to the market in the minimum possible time (Cheng et al., 1998).

One way in which manufacturing industries can take advantage of their agility is to form supply chains. Supply chains are ephemeral organizations in which several companies collaborate to produce a single product or product line. Participating in supply chains allows an agile company to use its knowledge, resources, and particular manufacturing expertise to take advantage of business opportunities that exist on a larger scale than the company could handle alone. To facilitate the creation of supply chains, potential partners must be able to quickly evaluate whether it will be profitable for them to participate in a proposed supply chain. Simulation technology in general, and distributed simulation technology in particular, can be used to enable the evaluation process. Each partner can use a simulation of its facilities to determine whether it has the capability to perform its individual function in the supply chain. Then, these simulations can be integrated into a distributed simulation of the complete supply chain, and used to predict viability and profitability of the proposed product collaboration. Use of distributed simulation technology allows each potential partner to hide any proprietary information in implementation of the individual simulation, but still provide enough information to evaluate the supply chain as a whole.

In this paper, a prototype for a distributed simulation that could be used to evaluate the viability of a supply chain will be described. The supply chain is comprised of several component manufacturers, final assembly plants, transportation systems, and warehouses. Potential information flows and material flows between supply chain members will be described.

Finally, a strategy for implementing a reusable simulation of a component manufacturer using commercially available simulation tools will be described in detail.

A sample supply chain to be used for the prototype distributed simulation system will be described in Section 2. The IDEFØ functional models among models in the distributed system will be presented in Section 3.1. Section 3.2 illustrates the time-sequence interactions between various members. Section 3.3 then shows the Deterministic Finite State Automata graph showing the behavior of various members. Details regarding the design, development, implementation and demonstration of the distributed simulation system are also made available in Section 4. Formal models described in Sections 3.1, 3.2, and 3.3 will form the basis for the work described in Section 4. A description of testing of the system and results inferred is discussed before we conclude the paper.

2. SUPPLY CHAIN SCENARIO

This section presents a prototype supply chain for manufacturing a product. Supply chain configurations will differ depending on production requirements and characteristics of potential collaborating companies. The prototype supply chain considered in this paper is composed of an assembly, two suppliers and a transportation system (see Figure 1). Note that the same formal modeling tools and techniques proposed in this paper can be used for differently configured supply chains. Four simulation models have been built to represent each of the above-mentioned players. These players interact with each other through technology that will be described in Section 4. The Assembly produces final products by assembling part A and part B. Part A is produced by Supplier A, and part B is produced by Supplier B. Assembly maintains a stock of the parts. When the stock falls below the prescribed threshold, it sends an order to the corresponding supplier for more parts. The roles of individual members of the supply chain are as follows:

- Parts Suppliers (Supplier A and Supplier B)
  Suppliers manufacture parts as necessary and use the Transporter to deliver manufactured parts to Assembly.
- Assembly
  Assembly puts together the final product by using the components sent by Parts Suppliers.
- Transporter
  Transporter moves parts from Parts Suppliers to Assembly.

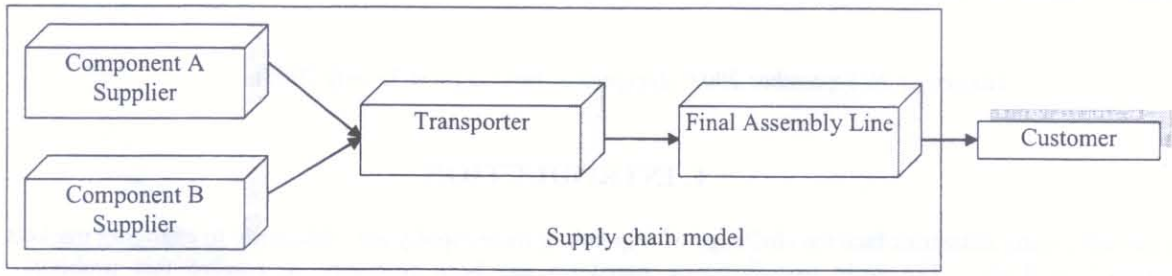


Figure 1. Prototype supply chain model

This supply chain system depicted in Figure 1 is a pull system. The assembly line maintains a buffer stock of both of components. When the stock of either component falls below the prescribed threshold level, a purchase order is issued to the corresponding supplier. Suppliers also continuously maintain a minimum level of stock to ensure that the assembly line always gets its requirements immediately. Interactions between the assembly line and the suppliers are initiated by the assembly line only. A 'handshake' interaction is performed to open as well as close a transaction.

3. MODELING SUPPLY CHAIN

3.1 Functional Modeling using IDEFØ

A functional model is a structured representation of the information and objects that define a correspondence between the activities occurring in a manufacturing system (Mayer, 1992). The primary focus of function modeling is on information and objects that create a unique occurrence of a given activity (Mayer, 1992). In this section, information flows as well as material flows among activities are identified using the IDEFØ formal function modeling method. The IDEFØ method has been used for modeling functions in an organization or a system and relationships between those functions (Mayer, 1992).

The function of the supply chain is composed of three sub-functions as shown in Figure 2. Each of these sub-functions represents one or more members of the system. The function A1, Final Assembly plant, interacts with both internal members of the system, as well as external players that do not belong to the system. The function A2, Component Suppliers, interacts with internal members and external members. The function A3, Transportation system, interacts with only other internal members. Two external players interacting with the system are raw material suppliers and customers. Customers put demands on the system, and raw material suppliers provide the required materials to make products to satisfy those demands.

The IDEF∅ diagrams represent interactions that occur among various players. However, they cannot represent when and in what sequence those interactions occur. Therefore, the following sections will present time-sequence diagrams and finite state automata for the supply chain.

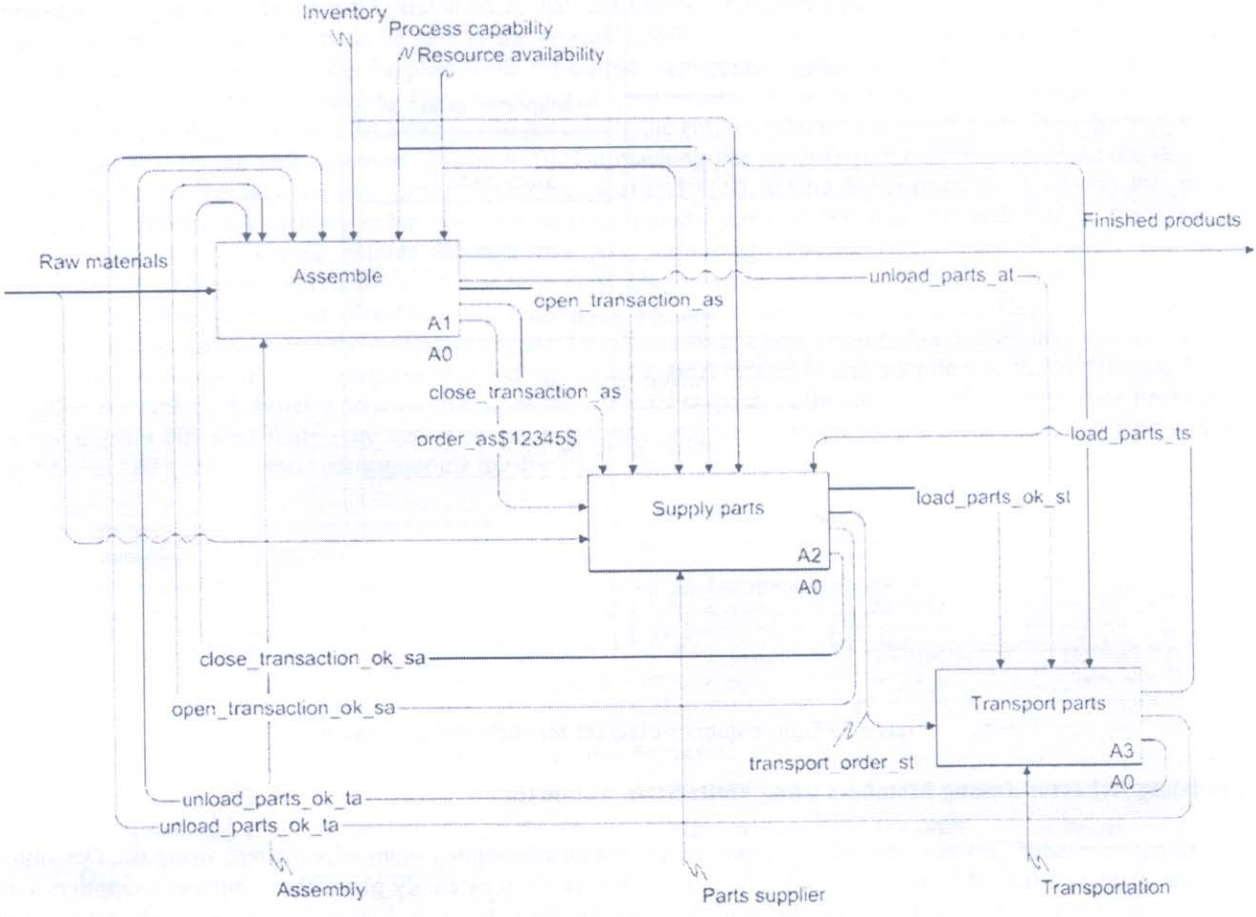


Figure 2. Second level decomposition of supply chain activities

3.2 Time Sequence Diagram

The sequence of interactions between Assembly, Supplier and Transporter is as shown in Figure 3. A sequence diagram has two dimensions: the vertical dimension represents time and the horizontal dimension represents different objects. An object's role in a Sequence diagram is shown as a vertical dashed line called the "lifeline". The lifeline represents existence of the object at a particular time. An object symbol is drawn at the head of the lifeline. A message is shown as a horizontal solid arrow from the lifeline of one object to the lifeline of another object. A message is communication between objects that conveys information with the expectation that action will ensue. Receipt of a message is similar to an event.

The assembler initiates interactions with the message *open_transaction_as* to the supplier. The supplier responds with the message *open_transaction_ok_sa*. The assembler's next message to the supplier is the *order_as12345*. The supplier then interacts with the transporter to load the parts. The supplier sends the message *transport_order_st* to transporter. The transporter then asks the supplier to *load_parts_ts*. The supplier confirms loading with the message *load_parts_ok_st*. The transporter then communicates with assembly to unload parts. The transporter sends the *delivery_order_*ta to the assembly. The assembly then asks it to *unload_parts_*at. The transporter confirms unloading by the message *unload_parts_ok_*ta. The

assembly, which initiated the transaction, also closes the transaction with that supplier, with the messages *close_transaction_as* and *close_transaction_ok_sa*.

This section described the interactions between one instance of the assembly model, supplier model and transporter model. It should be noted that the same approach could be extended for multiple instances of all of the objects. The description of activities or tasks carried out by each object between messages is presented in the next section using finite state automata.

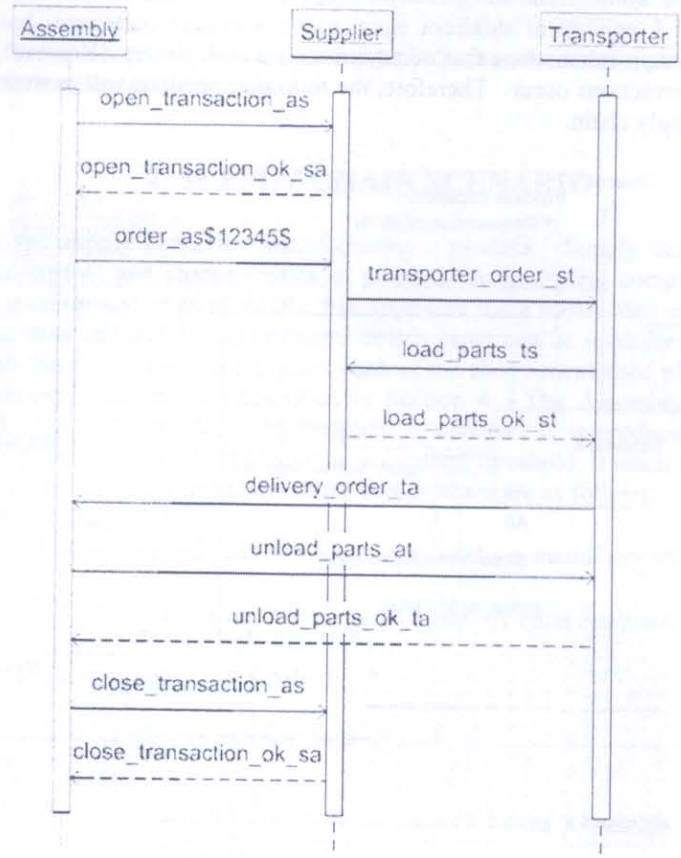


Figure 3. Time-sequence diagram for supply chain system

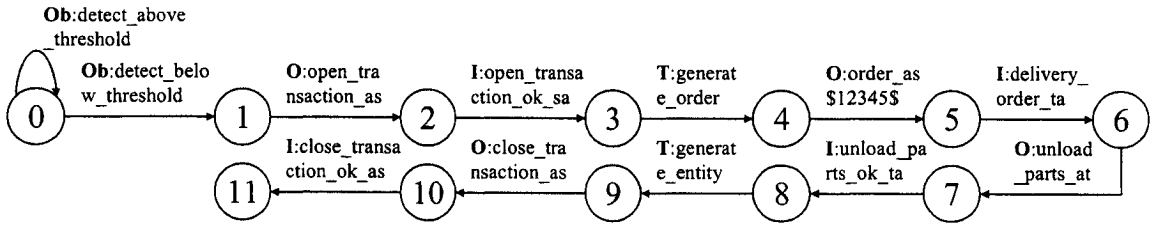
3.3 Modeling Behavior among Members using Finite State Automata

The coordination needed between components suppliers and the final assembly plant is performed using the Deterministic Finite State Automata (DFSA) (Martin, 1996). The DFSAs for the final assembly plant, the component suppliers and the transporter are shown in Figure 4. While the time sequence diagram shown in Figure 3 represents interactions among members as a whole, finite state automata graphs in Figure 4 represent behaviors within each individual member. Note that the two models basically convey the same information in different ways.

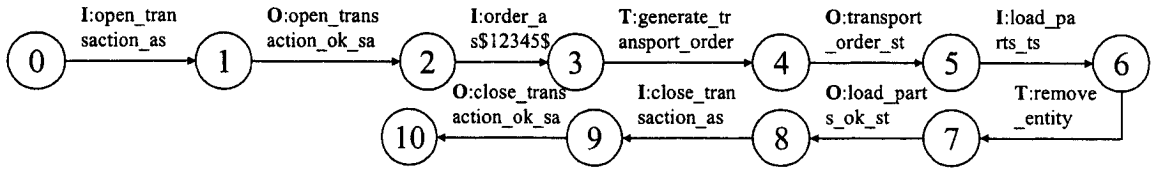
In Figure 4, circles with numbers indicate states or nodes. The arrows indicate action that on completion will allow the system to proceed to the next state. The actions need to be performed in the sequence shown. The "I", "O" and "T" in Figure 4 denote the incoming messages, outgoing messages and the tasks carried out, respectively. The "Ob" in Figure 4 denotes the observation carried out. In addition, messages ending with "as" denote messages from the assembly plant to a component supplier. Similarly, messages ending with "sa" denote messages from a component supplier to the assembly plant; messages ending with "at" denote messages from assembly to transporter; messages ending with "ta" denote messages from transporter to assembly; messages ending with "st" denote messages from component supplier to transporter; and messages ending with "ts" denote messages from transporter to supplier.

Initially, component suppliers, final assembly plant and transporter are at zero state (node). In this state, the final assembly plant observes or checks the quantity of component available, for every given time period. If the number is above the prescribed threshold (*detect_above_threshold*), it remains in the same state. If the number of components falls below the threshold (*detect_below_threshold*), it moves to the next state. Once it has reached state 1(node 1), it will not check the quantity of components again until the entire transaction is completed and it returns to zero state. The final assembly initiates the transaction between it and the supplier by sending the message *open_transaction_as* (Figures 4(a) and 4(b)). It then waits for the response, *open_transaction_ok_sa* (Figures 4(a) and 4(b)). Upon receiving this message it generates a

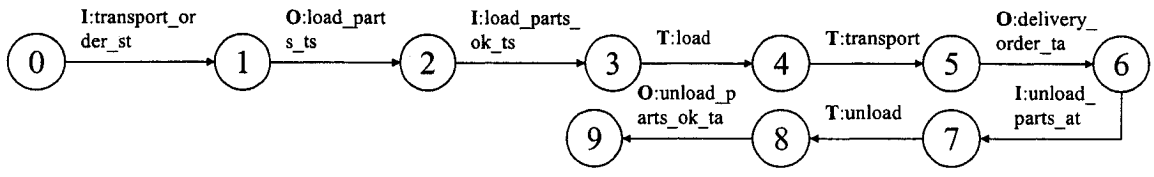
purchase order specifying component details and supplier details. Component details include information such as the component ID, component name, quantity required, price, expected delivery date, etc. Supplier details include information like the supplier name, supplier address, etc. For the current prototype under development, information used for component details includes component name, component ID, and quantity required; information used for supplier details includes the supplier name. The above purchase order generation is represented by the task *generate_order* (Figure 4(a)). After successful generation of the purchase order, the final Assembly plant sends the message *order_as12345* (Figures 4(a) and 4(b)). The number enclosed by the $ signs denotes the purchase order ID. The Supplier, on receiving the message, seizes the purchase order based on the number provided. The Supplier then generates a transport order, represented by the task *generate_transport_order* (Figures 4(b) and 4(c)), for sending the quantity of components requested. It is assumed that that quantity is always available immediately. It then sends the transport order, represented by the message *transport_order*_st (Figures 4(b) and 4(c)), to the Transporter. The Transporter on receiving the transport order, send back the message *load_parts_ts* (Figures 4(b) and 4(c)), to that Supplier, asking the Supplier to load the parts into the truck. The supplier then removes the quantity to be sent to the assembly from the buffer, shown by the task *remove_entity* (Figure 4(b)). After the task is complete, the Supplier sends the output *load_parts_st_ok* (Figures 4(b) and 4(c)), indicating to the Transporter that part loading is complete. Upon receiving this confirmation from the Supplier, the Transporter simulates the task *loading* (Figure 4(c)) through a time delay, also generating the entities. Then it transports parts from the Supplier to the Assembly, denoted by the task *transport* (Figure 4(c)). On reaching the Assembly, the Transporter submits the delivery order, shown by the message *delivery_order_ta* (Figures 4(a) and 4(c)), asking for permission to unload the parts. The Assembly on receiving the delivery order sends the message *unload_parts_at* (Figures 4(a) and 4(c)) to the Transporter. The Transporter then removes entities through the task *unloading* (Figure 4(c)). Then it sends the message *unload_parts_ok_ta* (Figures 4(a) and 4(c)) to the Assembly. The Assembly now knows that parts have been unloaded. So it generates the specified number of entities, represented by the task *generate_entity* (Figure 4(a)). After generating the components, the Assembly closes the transaction by sending the message *close_transaction_as* (Figures 4(a) and 4(b)). The response *close_transaction_ok_sa* (Figures 4(a) and 4(b)) returns final assembly and suppliers to the initial state. Note that the supplier remains in its initial state until it receives the initial message from the final assembly plant. The final assembly plant maintains a different finite state automata graph for each supplier. The messages are differentiated by adding the suffix "#1" or "#2", the numbers corresponding to suppliers.



(a) Finite state automata for assembly



(b) Finite state automata for component supplier



(c) Finite state automata for transporter

Figure 4. Finite state automata for supply chain system

4. DISTRIBUTED SIMULATION

This section presents an overview of distributed simulation to be used to evaluate potential supply chain systems. Again, the use of distributed simulation technology allows each potential partner to hide proprietary information about internal workings of a simulated system, but still provide enough information to evaluate the supply chain as a whole. The formal models presented in Section 3 have formed the basis for design and development of the distributed simulation system. A distributed simulation can be seen as a simulation that is comprised of multiple software processes that are independently executing and interacting with each other. The Department of Defense's High Level Architecture (HLA) (Kuhl et al., 1999; Fujimoto, 1998) for modeling and simulation can certainly be regarded as the state of the art in distributed simulation. The HLA establishes common high-level simulation architecture to facilitate interoperability of all types of models and simulations. The Run-Time Infrastructure (RTI) software implements the specification. It provides services in a manner that is comparable to the way a distributed operating system provides services to applications.

Figure 5 illustrates the relationships between the components of the distributed manufacturing simulation execution environment. An HLA-based simulation is called a *federation* (Kuhl et al., 1999). Each simulator that is integrated by the HLA RTI is called a *federate* (Kuhl et al., 1999). One common data definition is created for domain data that is shared across the entire federation. It is called the *federation object model* (FOM) (Kuhl et al., 1999). Note that each simulation model can be a legacy simulation system, and the models can be implemented in different languages (e.g., Arena™, AutoMod™, ProModel™, etc). The direct interaction of the simulation federates with the Runtime Infrastructure is quite complex and cumbersome. NIST has developed Distributed Manufacturing Simulation (DMS) Adapter to provide mechanisms for distributed simulation similar to those provided by the HLA RTI, but with a level of complexity that is manageable by the development resources available in the manufacturing community (Riddick and McLean, 2000). The interface of the adapter with simulation applications like Arena™ and ProModel™ will be discussed in upcoming sections.

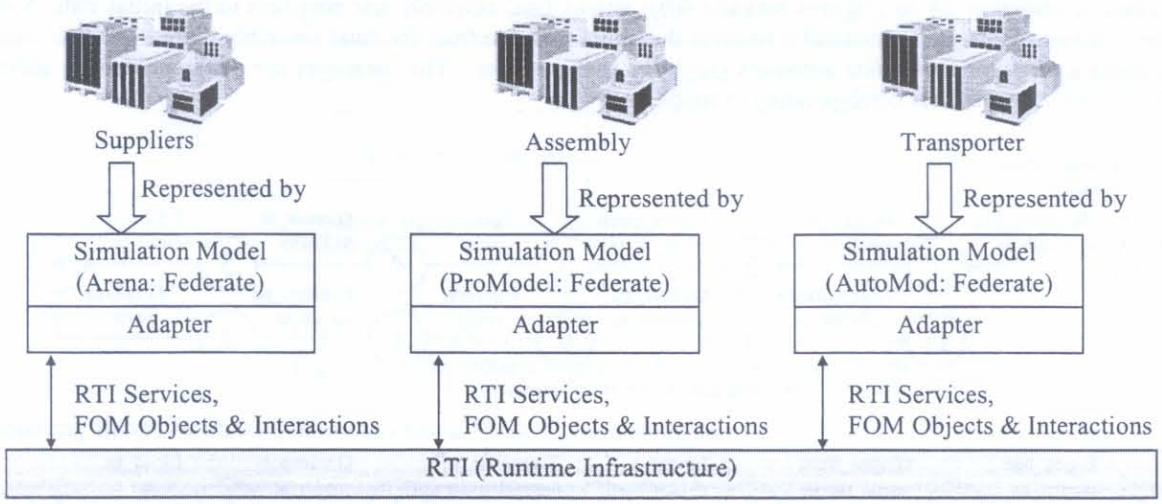


Figure 5. HLA based simulation integration architecture

The interactions among the members of the supply chain - the assembly, parts suppliers and the transporter, have been implemented using standard commercial simulation packages. In total, four simulation models have been created, one each for Assembly, Supplier A, Supplier B and Transporter (refer to Section 2 for the scenario). "Assembly", "Supplier B", and "Transporter" have been modeled using Arena™ 5.0, and "Supplier A" has been modeled using ProModel™ 4.0.

4.1 Modeling using Arena™

A generic interface module has been developed both for Arena™ and ProModel™ so that simulation models in these languages can interface with the RTI and the adapter. Since the developed interface modules are generic, the same modules have been used for component suppliers and the final assembly plant, with minor customizations. The simulation model can be broadly classified into 2 parts: the time management part (interface module) and the actual model.

The logic used in the time management part is shown in Figure 6. Note that the same concepts can be used when implementing the model using other discrete event simulation packages. One entity is created at zero time, and it is in charge of interface with the RTI and the adapter. As soon as it is created, it invokes a procedure, and delays for an amount

of time determined by the procedure. After delaying for the specified amount of time, it invokes the same procedure again. The entity repeats this process until the simulation is terminated. The pseudo code contained in the procedure is also shown in Figure 6. The first "if" condition checks whether the time of the local simulation is behind the current time of the global distributed simulation. If this gap is larger than the simulation step size (S_i), then the procedure advances the local simulation time by S_i. If the gap is smaller than S_i, then it advances the local simulation time by the amount of the gap. In the latter case, the local simulation time becomes equal to the global distributed simulation time. Note that time advancement in the local simulation is performed by specifying "a_time" value and delaying the simulation for "a_time" amount of time. If the simulation advance request from the local simulation has not been completed, the procedure halts the local simulation until it is completed. In other words, the local simulation needs to wait physically until all the other legacy simulations within the same federation catch up to the current time of the global distributed simulation.

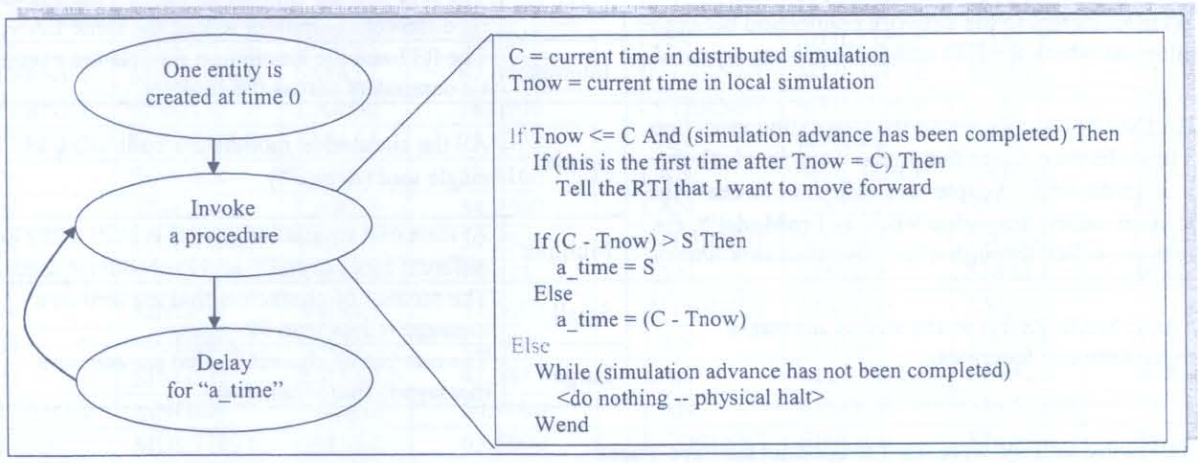


Figure 6. Logic used to interface with adapter and HLA

As discussed in earlier sections, the coordination needed between the two component suppliers and the final assembly plant is performed using the deterministic finite state automata (DFSA). Using global variables and arrays, the DFSA has been implemented. The procedure handling the message transactions is also contained in the procedure in Figure 6. However, due to limited space, the detailed procedure is not presented in this paper. The developed procedure is a generic procedure designed to handle any interaction between different companies, not restricted to the prototype discussed here.

4.2 Modeling using ProModel™

The "Supplier A" model has been built using ProModel™ 4.0. The same logic or algorithm (see Figure 6) specified for Arena™ is used to implement the Supplier model in ProModel™. In Arena™, a Visual Basic Application™ is used to implement the time and message management logic. For ProModel™ models, the time and message management is implemented using a dynamic link library (DLL) that has been compiled from C++ code. This DLL is interfaced with the ProModel™ model with the help of the 'External Files' feature of ProModel™.

4.3 Demonstration

Four simulation models (three models using Arena™ and one model using ProModel™) residing on different local area networks have run and interacted successfully. When the run begins, each model initializes with the RTI and a federation is created consisting of all four simulation models. Each model provides a message box after it initializes, to indicate that it is ready to advance to the running state. Only after the message box has appeared for every model in the supply chain can the models run simultaneously and interactively. Once the simulation completes, the models resign from the federation and the federation is destroyed in the RTI. Some important criteria to be considered in evaluating potential supply chains may include cost, quality of products with respect to manufacturers' capabilities, delivery times, and so on. Application of activity based costing (ABC) (Cooper and Kaplan, 1992) for cost analysis within the supply chain is left for future research.

4.4 Performance Testing

Preliminary performance tests have been conducted to evaluate the suitability of the proposed system in the Internet Environment. This prototype system developed has sufficient interactions among its different players to provide

researchers with data for analysis of its performance. The performance measures (responses) taken into account include 1) time taken by the simulation models to initialize, 2) time taken to transition from initialization state to running state, 3) time taken to advance the simulation time, 4) time taken to send messages and 5) time taken to terminate its connection. The factors determined to affect the system's performance are 1) the type of network, 2) the type of modeling tool used to build the simulation models and 3) the size of the messages exchanged between the models. The factors and levels are summarized in Table 1.

Table 1. Factors and their levels considered in the experiment

Factors (with description)	Levels	Description
NETWORK: Refers to the network connection between computers on which the RTI and federates are executed.	LAN	The RTI and the simulation models are executed on different computers within the same LAN.
	Internet	The RTI and the simulation models are executed on computers across the Internet.
MODELING TOOL: Refers to the simulation modeling tool. The difference arises due to the ways in which the functions in the DMS Adapter are called. In Arena™ the functions are called through a VBA. In ProModel™, the functions are called through a C++ dynamic link library.	Single	All the simulation models are built using a single tool (Arena™).
	Multiple	At least one simulation model is built using a different tool (Arena™ and ProModel™ used).
MESSAGE SIZE: Refers to the size of messages exchanged between federates.	Small	The number of characters that are sent as a message is less than 30.
	Large	The number of characters that are sent as a message is more than 300.

The following settings were used to conduct the experiment:

- Computer Type 1: Pentium III, 933MHz, Windows NT 4.0 Operating System.
- Computer Type 2: Pentium III, 933MHz, Windows 2000 Operating System.
- For all test runs with modeling tool factor = Single, all models were built using Arena™.
- For all test runs with modeling tool factor = Multiple, the Supplier A model was built using ProModel™, and the other models were built using Arena™.
- The HLA RTI (server) was always run on a Type 1 computer at The University of Arizona.
- For test runs within a LAN where modeling tool factor = Single, the simulation models were run on different Type 2 computers at The University of Arizona. Whenever a model used two simulations, the simulations were run on different computers.
- For test runs within a LAN where modeling tool factor = Multiple, the Supplier A model was run on the same Type 1 computer as the RTI. The other models were run on different Type 2 computers. Whenever a model used two simulations, the simulations were run on different computers.
- For test runs across the Internet where modeling tool factor = Single, the Supplier A model was run on a Type 2 computer at Arizona State University. The other three simulation models and the RTI were run on a single Type 1 computer at The University of Arizona across the Internet.
- For test runs across the Internet where modeling tool factor = Multiple the Supplier B model was run on a Type 2 computer at Arizona State University. The other three simulations models and the RTI were run on a single Type 1 computer at The University of Arizona across the Internet.

A total of 2^3 (three factors, two levels each) × 5 (five replications) test problems were generated by a permutation of the selected factors and their levels. The results of this test are shown in Table 2. Only the data collected for Supplier A is shown. The response shown in Table 2 is for the performance measure "time taken to send messages". For each problem, the response reported is the average of 11 observations made within each replication. The full factorial fit as calculated by Minitab™ for the response parameter and the three factors as independent parameters is shown in Tables 3 and 4. The level of significance (α value) selected is 0.05 for the analysis. The analysis of variance table (Table 3) gives a summary of the main effects and interactions. A p-value less than the α value (0.05) indicates that the corresponding effect is significant. Table 3 shows that the main effects, two-way interaction and the three-way interaction are all significant. The estimated effects and coefficients (Table 4) are calculated to evaluate the strength of the main effects and the interaction effects. The three-way interaction effect (network*modeling tool*message size), the two-way interaction effects (network*modeling tool and network*message size) and the main effects (network, modeling tool and message size) are all significant. Also, looking at the value of the effects, we determine the relative strength of each of the effects. Network has the greatest effect (20.568) on the time taken to send messages, the three-way interaction has the second greatest effect (10.015), and so on.

Table 2. Results of experiment, showing the average time taken to send messages

Network	Modeling tool	Message size	Avg. time taken to send message in ms
INTERNET	SINGLE	SMALL	69.0909
LAN	MULTIPLE	LARGE	46.6666
LAN	SINGLE	LARGE	55.0000
INTERNET	MULTIPLE	LARGE	74.1667
INTERNET	SINGLE	LARGE	61.9090
LAN	MULTIPLE	LARGE	36.4545
INTERNET	MULTIPLE	SMALL	87.8462
INTERNET	SINGLE	LARGE	38.0909
INTERNET	MULTIPLE	SMALL	75.8333
LAN	SINGLE	LARGE	58.4166
LAN	SINGLE	LARGE	54.2500
LAN	MULTIPLE	LARGE	39.2500
INTERNET	MULTIPLE	SMALL	78.3333
LAN	SINGLE	SMALL	37.5000
LAN	MULTIPLE	SMALL	53.5000
LAN	SINGLE	SMALL	42.5833
INTERNET	SINGLE	LARGE	50.9090
INTERNET	MULTIPLE	SMALL	63.5000
LAN	MULTIPLE	SMALL	41.8333
INTERNET	MULTIPLE	LARGE	76.0000

Network	Modeling tool	Message size	Avg. time taken to send message in ms
INTERNET	MULTIPLE	LARGE	71.8333
INTERNET	MULTIPLE	LARGE	78.4167
LAN	MULTIPLE	SMALL	60.2500
LAN	MULTIPLE	SMALL	55.8333
INTERNET	SINGLE	SMALL	59.2727
LAN	MULTIPLE	LARGE	47.2727
INTERNET	MULTIPLE	SMALL	89.2500
INTERNET	MULTIPLE	LARGE	82.0833
LAN	SINGLE	LARGE	53.9090
LAN	MULTIPLE	LARGE	43.3333
LAN	SINGLE	SMALL	42.8181
INTERNET	SINGLE	LARGE	61.9166
LAN	SINGLE	SMALL	46.6666
INTERNET	SINGLE	SMALL	75.1667
LAN	SINGLE	LARGE	51.8333
LAN	MULTIPLE	SMALL	60.9090
LAN	SINGLE	SMALL	47.5833
INTERNET	SINGLE	SMALL	75.1966
INTERNET	SINGLE	LARGE	43.6767
INTERNET	SINGLE	SMALL	74.7272

Table 3. Analysis of Variance for response (time taken to send messages)

Source	DF	SS	MS	F	P	Significant at α=0.05
Main Effects	3	5202.16	1734.05	35.66	0.000	Yes
2-Way Interactions	3	1049.91	349.97	7.20	0.001	Yes
3-Way Interactions	1	1002.95	1002.95	20.62	0.000	Yes
Error	32	1556.10	48.63			

Table 4. Estimated effects and coefficient for response (time taken to send messages)

Term	Effect	Coefficient	Standard Error of Coefficient	T	P	Significant at α=0.05
Constant		59.077	1.103	53.58	0.000	
Network	20.568	10.284	1.103	9.33	0.000	Yes
Modeling	8.102	4.051	1.103	3.67	0.001	Yes
Message	-5.615	-2.808	1.103	-2.55	0.016	Yes
Network*Modeling	8.628	4.314	1.103	3.91	0.000	Yes
Network*Message	-5.306	-2.653	1.103	-2.41	0.022	Yes
Modeling*Message	-1.546	-0.773	1.103	-0.70	0.488	No
Network*Modeling*Message	10.015	5.007	1.103	4.54	0.000	Yes

An experimental approach similar to the one detailed above was used to analyze the other performance measures. The significance of the different factors' effects on the various performance measures and their interactions are summarized in Table 5. Network is found to be a significant factor for all the performance measures. This indicates that a good network connection will improve the performance of distributed simulation. With the current growth of technology, higher speed communication lines will make network a less significant factor. The statistical significance of the message size factor can be curbed by using standard, consistent and concise messages throughout the distributed simulation. Finally, other

interesting observations made during the experiment include: 1) 95% confidence interval for the time taken to initialize when tested within the LAN is 6.0 – 6.8 seconds, 2) 95% confidence interval for the time taken to initialize when tested across Internet is 12.7 – 24.4 seconds, and 3) the time taken to perform the different Adapter functions during the first replication is not significantly different from the time taken by the rest of the replications.

Table 5. Factors having significant effect on the different performance measures

Performance measure	Significant Factors (in order from top to bottom)
Time taken to initialize	Network (LAN takes lesser time)
Time for transition from initial state to running state	Network (LAN takes lesser time)
Time taken to advance simulation	Two-way interaction of Network*Modeling tool Modeling tool (Single takes lesser time) Network (LAN takes lesser time)
Time taken to send messages	Network (LAN takes lesser time) Three-way interaction of Network*Modeling tool*Message size Two-way interaction of Network*Modeling tool Modeling tool (Single takes lesser time) Message size (Small takes lesser time) Two-way interaction of network and message size
Time taken to terminate simulation	Network (LAN takes lesser time)

5. CONCLUSION

In this paper, a prototype for a distributed manufacturing simulation was described. First, the information flow as well as material flow has been modeled using the IDEFØ, a formal modeling tool. Second, the sequence of interactions has been represented using the time sequence diagram and the finite state automata. These formal models have formed the basis for the development of the proposed distributed simulation system. Third, mechanisms have been described to govern time management and communication between the member simulations in a distributed simulation. Fourth, these mechanisms have been implemented and demonstrated using commercially available simulation packages. Fifth, some factors affecting the performance of the distributed simulation have been identified. Tests were conducted and the data analyzed using statistical software. Factors such as network, software used and size of messages exchanged were found to be significant. However, with the current pace of technology growth and with the use of standardized tools, these factors can be made less significant. Thus, based on the preliminary results obtained, it is found that distributed simulation technology is viable to analyze complex, globally dispersed supply chains. This technology will help gain a deeper insight about supply chain system behavior, which in turn will lead to better decision making throughout the supply chain.

6. REFERENCES

1. Cheng, K, Harrison, D. K., and Pan, P. Y. (1998). Implementation of agile manufacturing – an AI and Internet based approach. Journal of Materials Processing Technology, 76: 96 - 101.
2. Cooper, R., and Kaplan, R. S. (1992). Activity-based systems: measuring the costs of resource usage. Accounting Horizons, September, pp. 1-13.
3. Fujimoto, R. M. (1998). Time management in the High Level Architecture. Simulation, 76:6, pp. 388 - 400.
4. Kuhl, F., Weatherly, R., and Dahmann, J. (1999). Creating Computer Simulations: An Introduction to the High Level Architecture. Prentice-Hall, Upper Saddle River, NJ.
5. Martin, J. C. (1996). Introduction to Languages and the Theory of Computation. McGraw-Hill, 2nd Edition, New York.
6. Mayer, R. J. (1992). IDEFØ Function Modeling Method Report, Knowledge Based Systems Inc., College Station, TX.
7. Riddick, F., and McLean, C. (2000). The IMS Mission architecture for distributed manufacturing simulation. Proceedings of the 2000 Winter Simulation Conference, Orlando, FL, December 10-13.