Proceedings of the 2001 Winter Simulation Conference
B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer, eds.

TOWARDS COTS DISTRIBUTED SIMULATION USING GRIDS

Simon J.E. Taylor
Rajeev Sudra
Tharumasegaram Janahan

Centre for Applied Simulation Modelling
Brunel University
Uxbridge UB8 3PH, UNITED KINGDOM

Gary Tan

School of Computing
National University of Singapore
Kent Ridge 119260, SINGAPORE

John Ladbrook

Dunton Engineering Centre
Laindon, Basildon
Essex SS15 6EE, UNITED KINGDOM

ABSTRACT

This paper reports on continuing work that concerns research into the development of a commercial off the shelf (COTS) distributed simulation environment (federation) using the Generic Runtime Infrastructure for Distributed Simulation to support the interoperation of simulation packages such as Arena, Extend, Simul8, Taylor, Witness, etc. The main aim of this work is to provide the industry with a business benefit from distributed simulation by making it possible to reuse previously developed models in order to address different problems within enterprises or between enterprises (supply chains) that could not otherwise be addressed due to barriers of cost and time. The approach emphasises transparency and minimal intervention with the simulation modeller. Two cases are presented: a distributed supply chain simulation (federation), and an example from the automotive industry.

1 INTRODUCTION

This paper reports on continuing work that concerns research into the development of a commercial off the shelf (COTS) distributed simulation environment for industries that currently use COTS discrete event simulation packages that are available today (Arena, Extend, Simul8, Taylor, Witness, etc.). The main aim of this work is to provide industry with a business benefit from distributed simulation by making it possible to connect and reuse (Pidd, Oses and Brooks 1999) previously developed models in order to address different problems within enterprises or between enterprises (supply chains) that could not otherwise be addressed due to barriers of cost and time. In addition to this, in order to maximise the benefit of distributed simulation to industry while minimising the cost of the use of this technique, the overall philosophy of this research is *transparency*; distributed simulation is to be used with as few additional skills required by the simulation modeller and (ideally) no technology-specific additions to simulation methodology.

This work is based on the *Generic Runtime Infrastructure for Distributed Simulation* (GRIDS) (Taylor, Saville, and Sudra 1999), an extensible middleware for research into distributed simulation tools and techniques. Instead of the fixed service groups advocated for a runtime infrastructure by the HLA, GRIDS provides basic services for the interoperation of federates within a federation. Extensibility is provided by Thin Agents that encapsulate additional services as and when required (dead reckoning, time management, message filtering, security, etc.). This research is intended to complement other research carried out in this area: Zeigler, Kim, and Buckley (1999) investigate the use of DEVS for the interoperation of models, Turner, Cai and Gan 2000 and Gan et al. 2000 investigate the use of the DMSO RTI for distributed supply chain simulation, and the MISSION Project (McLean and Riddick, 2000) investigate the use of the DMSO RTI within the context of manufacturing systems integration.

Three GRIDS-based projects are currently underway. The first concerns the continued development of a Distributed Supply Chain simulation, the GRIDS Supply Chain Federation (GRIDS-SCF), that builds on work previously reported in Sudra, Taylor, and Janahan (2000a; 2000b). The goals of this work are to investigate problems involved in

the development of a generalised COTS distributed simulation, and to create a demonstrator that can be used to communicate the various advantages (and complications) of using distributed simulation in various industrial contexts. The second and third projects involve the development of distributed simulation-based solutions in the automotive industry (with the Ford Motor Company) and in business process simulation (with British Telecommunications). In this paper we report on the progress of the first two of these projects. Space limits discussion to that of time management and interconnection. Other concerns such as distributed model validation, distributed experimentation methods, performance, network security, and hiding of proprietary information are also under consideration. For example Tan and Taylor (Taylor, Tan and Ladbrook, 2001) discuss the implications of this work with regard to Data Distribution Management and the Object Model Template.

This paper is structured as follows. In section 2 we introduce GRIDS-SCF, our demonstration supply chain federation that is used to investigate and to communicate issues involved in COTS distributed simulation. Section 3 presents the technique by which our RTI (GRIDS) supports model interoperability with a COTS package. We then present the problem specified by Ford that we are attempting to use distributed simulation to investigate (and our GRIDS-based solution). We end the paper with some conclusions in section 5.

2 GRIDS-SCF: A DISTRIBUTED SUPPLY CHAIN SIMULATION

The goals of the first of our three COTS distributed simulation projects are to develop a hypothetical supply chain simulation that allows us to investigate the development of a generic approach to COTS distributed simulation, and to create a bridge between simulation concepts found in industries that use COTS simulation environments (manufacturing, telecommunications, service industries for example) and simulation concepts used in distributed simulation. The latter of these goals is important as there is a gap between the terminology, concepts and methods of these two areas of simulation and may prove to be a significant barrier to the adoption of distributed simulation technology. There has already been some success in this approach as it has proved to be the inspiration of the other two GRIDS COTS projects.

The supply chain model that forms the basis for GRIDS-SCF is composed of three companies and is shown in Figure 1 (dashed boxes are intended to indicate the scope of the three models A, B and C). The motivation for simulating the entire supply chain is to determine if company C can be supplied in time to meet the demand for its new product that can be manufactured as a result of the new supply chain (virtual enterprise). Company A manufactures and supplies components for company B. Company B assembles these components, recycles any poor quality components back to company A, and passes the assembly on to company C for packaging and finishing (and selling). If company C finds that an unfinished assembly is of poor quality, the assembly is sent back to company B for reprocessing. Note the input point in model A (EntryPtA) and output point in model C (ExitPtC). These are constructs that are normally added to a model to control the introduction of relevant entities into the model and to record the exit of entities from the model. Individually each of the models would have other input and output points. These must be replaced if the model is to be integrated (subject to validation) and can require additional model features to be added (such as a buffer). Also note that these models are "high-level" and only include features that are pertinent to our discussion. A "real" model of a supply chain component would possess significantly more detail to reflect the workings of its particular systems. Note also (and importantly!) that these additions are due to the requirements of modelling and not distributed simulation.

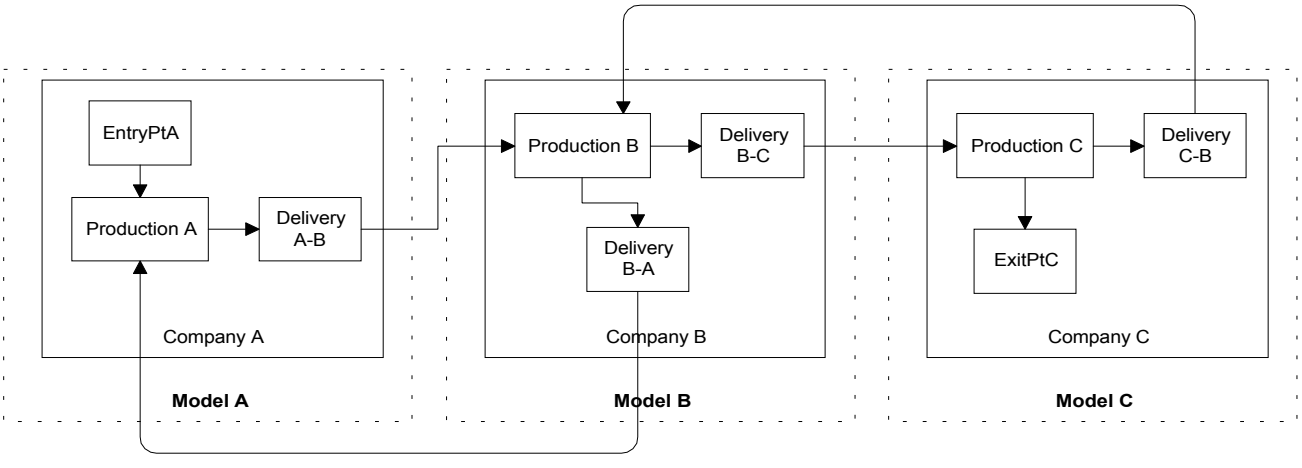

Figure 1: Supply Chain Model

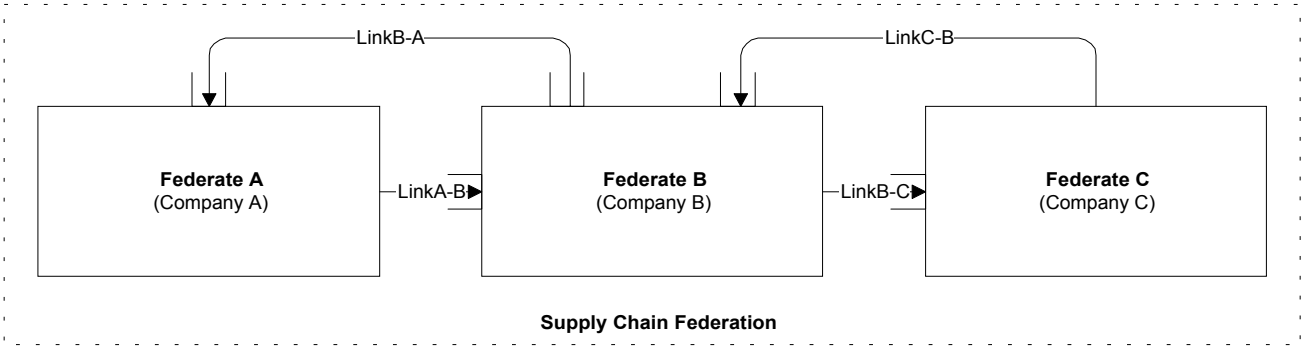

Figure 2: Supply Chain Federation

Conventionally, each of these models would be built using a COTS simulation environment. These are typically visual simulation environments that are used to graphically build the simulation model and are then (subject to verification and validation) used to experiment with the model. If we assume that each of the models of our hypothetical supply chain runs in a COTS environment running on different computers, then to realise a distributed supply chain simulation we must make some allowance for the transfer of information between models (computers). In HLA-style distributed simulation terminology, a model running on a COTS environment modified for distributed simulation can be called a *federate*. The collection of these that compose the supply chain simulation may be called a *federation* (hence Supply Chain Federation (SCF)). Information is shared between federates by sending time-stamped event messages to each other via software (middleware) called a runtime infrastructure (RTI). The timestamp of the event message indicates the time that a consignment is scheduled to arrive at a particular company. The details of the event message indicate the contents of the consignment (identification, volume, etc). Program code within the COTS package (i.e. event routines) must be modified to reflect this.

A requirement placed on any simulation is that, by the end of the simulation, all events will have been processed in ascending order of time. While this is intuitive (and well-known in the parallel and distributed simulation communities (Fujimoto 1999)), the consequence to distributed simulation is that each federate must correctly process all events that it schedules for itself with all events scheduled for it by other federates. Some kind of mechanism is required to synchronise the event messages sent to a federate with the events that it schedules for itself (we restrict ourselves to the set of solutions that do not use a global clock). One such mechanism is the conservative synchronisation protocol.

The conservative synchronisation protocol places rigid requirements on a federation. These are as follows. Each federate has its own simulation clock, event list, simulation executive, model, and event routines needed to execute each event and to send event messages (most of which is contained within the COTS simulation package within the federate). Each possible event message interaction between these federates is formalised as a *link* (Figure 2). In this figure our example supply chain model has become a supply chain federation. There are four links: linkA-B, linkB-A, linkB-C and linkC-B. At the end of each link is a *link queue* in which messages awaiting processing are queued (queueA-B, queueB-A, queueB-C and queueC-B respectively). Each link also has a *link clock* (clockA-B, clockB-A, clockB-C, and clockC-B). A link clock will either indicate the time of the message waiting in its queue or the time of the last message removed from it. To ensure that each federate (and therefore the entire federation) executes events in the correct order, a federate must augment its simulation executive with, for example, a conservative protocol with deadlock avoidance modified for inclusion in a federate. The topology of our SCF model owes much to the generalisation of an approach to synchronise between different federate.

2.1 Lookahead

Lookahead presents a major problem in distributed discrete event simulation. Lookahead is effectively the greatest simulation time that a federate can guarantee that nothing will occur in the model that it is simulating. If this guarantee is zero, then the conservative protocol cannot be used (a similar observation can be made of other protocols). In our model lookahead can be derived from the time calculations (typically probabilistic distributions) used to schedule the arrival of an entity at a federate. We shall term the value of lookahead assigned to a particular link *lookaheadX-Y*, where X-Y is the designation of the linkX-Y. There are several problems associated with the derivation of lookahead (such as the calculation of lookahead based on open ended probabilistic distributions or the existence of multiple time distributions). In summary, as the time-stamps of event messages sent between federates (via links) may be based on different distributions (associated with different activities within a federate's model), it is important to recognise this and reflect it in the value of lookahead assigned to that link. Failure to do so may im-

pact on performance or invalidate the protocol. Furthermore, the actual calculation of the value of lookahead for a particular link may be problematic. For example, if the distribution governing the timestamps of event messages is stochastic then there is no guarantee that zero might be sampled from this distribution; there is no guarantee that lookahead could not be zero. It is absolutely vital that whatever method is used for the calculation of lookahead it is reflected in the validation of the federation (i.e. no convenient fixes!). For the purposes of this discussion we shall assume deterministic, non-zero distributions are used (thus giving fixed lookaheads).

We shall now consider the implications of the above to a COTS simulation package.

3 COTS INTEGRATION

To continue our discussion of how COTS simulation environments may be adapted for distributed simulation, we now consider some of the requirements made on a COTS environment by a distributed simulation such as our SCF. We will limit our discussion to time management (distributed experimentation and event translation will be discussed in a later paper). Consider the conservative protocol discussed so far. As each COTS possesses its own event list, and distributed simulation dictates that events on the event list must be integrated with incoming event messages, there must be some method available to synchronise between processes handling event messages (link queues, etc.) and the simulation executive of the COTS.

In our work we use GRIDS to link the COTS simulation package federates and a Thin Agent within GRIDS to handle the time management (see Sudra et al. 2000a and Sudra et al. 2000b for more details on this approach). The particular Thin Agent used is currently the CPADS-TA (conservative parallel and distributed simulation-thin agent) although others are under development. The original form of this thin agent required that the event list of a COTS was completely removed. On reflection, however, the mechanism was modified to reduce the amount of alteration required of the COTS and to make the method of synchronisation close to that required by the HLA. Figure 3 shows the makeup of a federate. In our scheme a federate is composed of a GRIDS client and the modified COTS. Our time synchronisation requires the COTS to expose an interface (implementable via a socket) and to modify its simulation executive. The network interface manager of the GRIDS client is responsible for routing event messages to and from the federate, while the thin agent manager determines what message traffic is routed to and from the COTS simulation package, the CPADS-TA, and the network interface manager. The meta-database contains appropriate information for the various modules to function (such as the mapping of federates to machines). The thin agent and the contents of the meta-database are received from the GRIDS server on registration and initialization.

When the COTS package attempts to process the next event off its event list, it must ask GRIDS for permission (via a message from the COTS package to the GRIDS client). This message will be REQUEST_TIME_ADVANCE (current_time). The message will be routed via the thin agent manager to the CPADS-TA which will perform the conservative algorithm described above and respond with

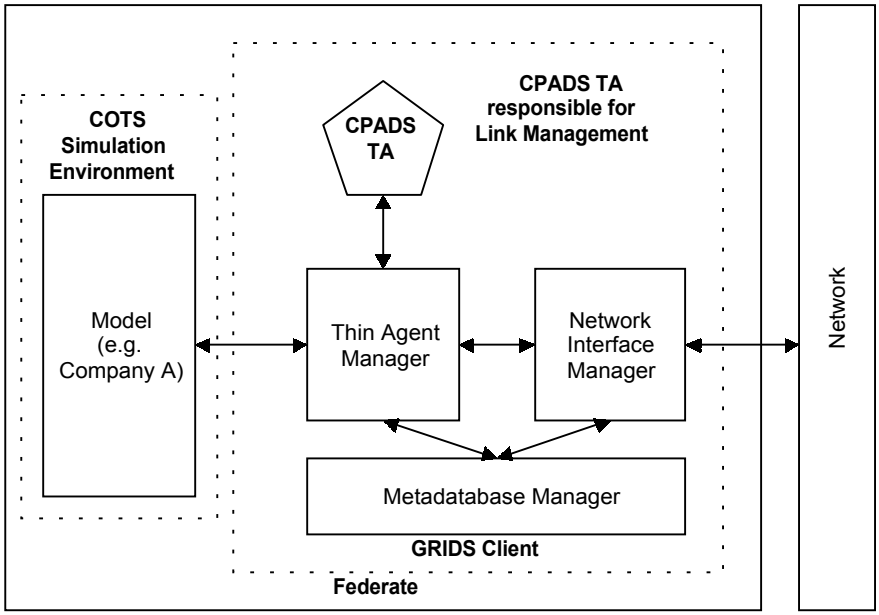

Figure 3: Federate Composition

either TIME or EVENT(time, type) messages. If the response is TIME, then the COTS package will advance its clock to that time. If the next event(s) are less than or equal to this new time then they may be processed. If the response is INPUT_EVENT(time, type), then an event message has arrived. Events off the event list are first processed up to the timestamp of the event message and then the event represented by the event message is processed. If, as a result of processing an event, the COTS package attempts to schedule an event effecting another federate (arrival of widget components, for example), the modified event routine code will generate an OUTPUT_EVENT(time, type) to the GRIDS client. GRIDS then forwards the message to the appropriate federate. OUTPUT_EVENT is used to indicate both events and null messages. Lookahead is calculated as indicated in section 2.1. The algorithms are as follows.

3.1 COTS Simulation Environment Algorithm

```
while not terminated
  determine next event time
  send REQUEST_TIME_ADVANCE(current_time) to
    GRIDS client
  wait for response
  if response is TIME then
    while (next event time < TIME)
      advance simulation clock to next event
        time
      process next event (send event
        messages, etc.)
    endwhile
  else (response is EVENT(time, type))
    while (next event time < EVENT(time))
      advance simulation clock to next event
        time
      process next event (send event
        messages, etc.)
    endwhile
    advance simulation clock to EVENT(time)
    process EVENT(type) (send event messages,
      etc.)
  endif
endwhile
```

3.2 GRIDS Client Algorithm (CPADS Thin Agent)

```
while not terminated
  wait for
  REQUEST_TIME_ADVANCE(current_time)from COTS
    if (REQUEST_TIME_ADVANCE(current_time)
    < link clock [i]) then
    TIME = REQUEST_TIME_ADVANCE
    send TIME to COTS
    else if minimum is link clock[i] then
      if link queue[i] has an event message
        then
        remove event message
        send EVENT(time, type) to COTS
      else if link queue[i] has a null
        message then
        remove null message
        TIME = timestamp of null message
        send TIME to COTS
      else if link queue[i] is empty then
        send null messages
        wait until a message arrives in a
            link
        queue then continue (COTS waits)
      endif
    endif
  endwhile
```

We now present an overview of the second of our three GRIDS COTS projects.

4 AUTOMOTIVE CASE STUDY

This work is being carried out with the Ford Motor Company and is typical of the model reusability problems encountered in industry. The goal here is to integrate several models that have been developed separately so that a new problem may be investigated conveniently.

Engine assembly is a complex problem involving the manufacture and assembly of a wide variety of parts into several possible engine types (different capacities, fuel injection options, etc.). The demand for different engines is determined by car orders. The demand for the different parts is ultimately derived from this, but, given that it takes time to machine the different possible parts, and production lines need to be reconfigured for each part production, buffer stock is required to keep the engine assembly from waiting. The problem that needs to be addressed is how much of each manufactured part must be kept in the buffer. Too little and engine production is stalled, too much and excess capital becomes tied up in stock.

Figure 4 shows the current configuration of the engine production system. As can be seen, engine production is essentially performed in two linked areas – machining and assembly. In machining there are five machining lines (head, camshaft, con rod, crank, and block), each of which is responsible for machining a particular range of engine parts. Each of the lines comprises a number of operations and a number of machines. These feed into the assembly area. Heads and camshafts are assembled in the cylinder head assembly area. The con rods feed into the piston and rod assembly. Outputs from these two assemblies and the remainder of the machining lines feed into a conveyor that contains pallets on which the engine blocks are built through a combination of automatic and manual operations. When finished the engine block passes through a hot test and after test dress operation which either passes the assembled engine onto the car plant or returns it for replacement of defective parts. Currently the system is modelled as six different models in the COTS simulation package Witness (Lanner Group, UK) (five machine line models and one assembly line model).

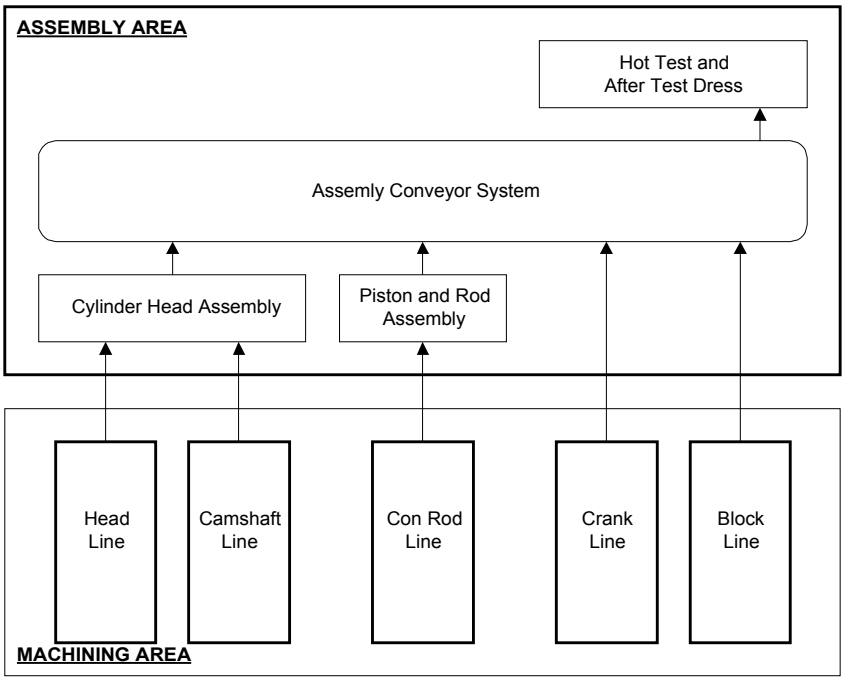

Figure 4: Engine Production System

4.1 A Distributed Simulation Solution

While it would be possible to re-code and integrate the model into one (we assume that the capability of the COTS package is extensive enough!), it would be preferable to connect the models together via distributed simulation. This discussion assumes that there exists a COTS simulation package that has been modified for distributed simulation. We now investigate the consequences of combining the six existing models into one federation.

The first problem that must be addressed is that regardless of whatever distributed simulation protocol is used the models themselves do not have the features required for connection. The head line, for example, is typical of a simulation model in that it has an input distribution that models the arrival of raw materials and an output that registers the departure of machined heads for statistical purposes. The assembly area model has five inputs and one output. To connect the models we must reconsider the system being modelled. Consultation with the simulation modeller led to several model modifications. This essentially introduces a new process *travel* that models the time taken to move parts from the machining lines to the assembly area, buffers to store the waiting stock, and *select*, another process that models the selection of buffered parts for the assembly area (this also contains the statistical routines required to calculate the scarcity of resources). This is shown in Figure 5.

Once this has been accomplished, the federation is implemented as shown in Figure 6. The topology of this distributed simulation is extremely simple as each of the five machine line federates is effectively a producer of event messages for the assembly area federate to consume. Lookahead is irrelevant in this case as deadlock is not possible (lookahead would be calculated on the basis of the travel times which, the simulation modeller has indicated, are deterministic). The event translation only effects the travel processes (one output event per machine line, five input events for the assembly area). GRIDS is responsible for the transport of event messages with the CPADS-TA responsible for time synchronisation (the CPADS-TA is essentially redundant in the five machine line federates).

5 CONCLUSIONS

This paper has introduced two of the three GRIDS projects that are being used to investigate how COTS simulation environments can be modified for distributed simulation with minimum intervention in simulation methodology. Demonstrations have been implemented in Java and are being used to investigate performance. The integration with COTS packages requires modification to the package and a socket interface. Negotiations with simulation vendors for this are on-going and are helped by the backing of end users who can see the business benefits of this transparent, low intervention approach. It is hoped that the work described in this paper contributes to the emerging use of distributed simulation in the industrial sector so that, eventually, the whole sector will commercially benefit from its use.

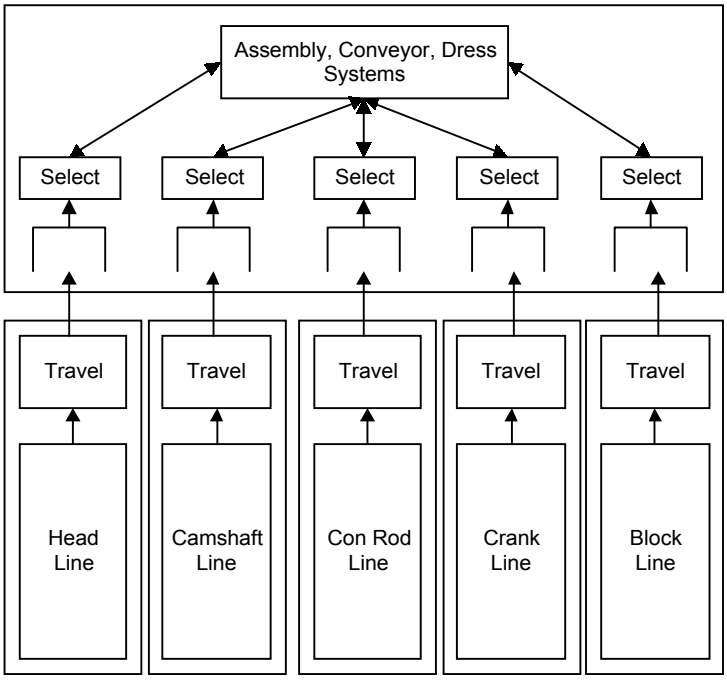

Figure 5: Model Modified for Integration

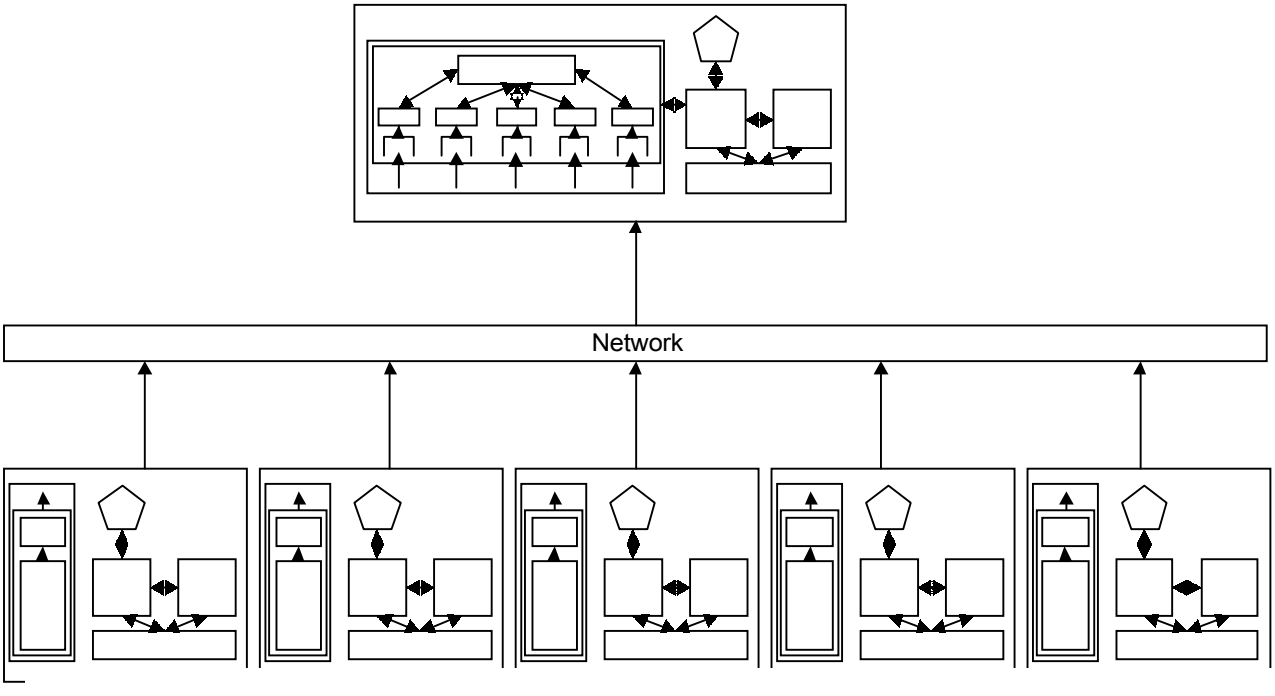

Figure 6: Automotive GRIDS Federation

ACKNOWLEDGMENTS

Thanks must go to Simon Dennis of British Telecommunications for his support in this work. This work is partially supported by the GROUPSIM technology programme network.

REFERENCES

Fujimoto, R.M., 1999. *Parallel and Distributed Simulation Systems* (Wiley Series on Parallel and Distributed Computing). John Wiley & Sons, New York, NY.

Gan, B.P., L. Liu, S. Jain, S.J. Turner, W. Cai, and W. Hsu. 2000. Distributed Supply Chain Simulation Across Enterprise Boundaries. In *Proceedings of the 2000 Winter Simulation Conference*. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds, 1245 – 1251. Orlando, FL.

McLean, C. and F. Riddick. 2000. The IMS Mission Architecture for Distributed Manufacturing Simulation. In *Proceedings of the 2000 Winter Simulation Conference*. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds, 1539-1548. Orlando, FL.

Pidd, M., N. Oses, and R.J. Brooks. 1999. Component-Based Simulation on the Web? In *Proceedings of the 1999 Winter Simulation Conference*. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, eds, 1438-1444. Phoenix, AZ.

Sudra, R., S.J.E. Taylor and T. Janahan. 2000a. Distributed Supply Chain Management in GRIDS. In *Proceedings of the 2000 Winter Simulation Conference*. J. A. Joines, R. R. Barton, K. Kang, and P. A. Fishwick, eds, 356 – 361. Orlando, FL.

Sudra, R., S.J.E. Taylor and T. Janahan. 2000b. GRIDS: A Novel Architecture for Distributed Supply Chain Management. In *Proceedings of the Fall 2000 Simulation Interoperability Workshop*. Orlando, FL. 00F-SIW-051.

Taylor, S.J.E., J. Saville, and R. Sudra. 1999. Developing interest management techniques in distributed interactive simulation using java. In *Proceedings of the 1999 Winter Simulation Conference*. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, eds, 518-523. Phoenix, AZ.

Taylor, S.J.E., G. Tan and J. Ladbrook. 2001. Data Distribution Management for Distributed Supply Chain Management. In *Proceedings of UKSIM'01, Conference of the UK Simulation Society*, David Al-Dabass, ed. 35-41. Cambridge, UK.

Turner, S.J., W. Cai and B.P. Gan. 2000. Adapting a Supply Chain Simulation for HLA. In *Proceedings of the 4th International Workshop on Distributed Simulation and Real Time Applications*. 71-78. IEEE Computer Society Press. San Francisco, California, U.S.A.

Ziegler, B.P., D. Kim, and S.J. Buckley. 1999. Distributed supply chain simulation in a DEVS/CORBA execution environment. In *Proceedings of the 1999 Winter Simulation Conference*. P.A. Farrington, H.B. Nembhard, D.T. Sturrock, and G.W. Evans, eds, 1333-1340. Phoenix, AZ.

AUTHOR BIOGRAPHIES

SIMON J.E. TAYLOR is the Chair of the Simulation Study Group of the UK Operational Research Society. He is a Senior Lecturer in the Department of Information Systems and Computing and is a member of the Centre for Applied Simulation Modelling, both at Brunel University, UK. He was previously part of the Centre for Parallel Computing at the University of Westminster. He has an undergraduate degree in Industrial Studies (Sheffield Hallam), a M.Sc. in Computing Studies (Sheffield Hallam) and a Ph.D. in Parallel and Distributed Simulation (Leeds Metropolitan). His main research interests are distributed simulation and applications of simulation health care. He is also a member of the Purple Theatre Company. His email and web addresses are <simon.taylor@brunel.ac.uk> and <www.brunel.ac.uk/~csstsjt>.

RAJEEV SUDRA is a Ph.D. candidate in the Department of Information Systems and Computing at Brunel University, UK. He received his B.Sc. in Computer Science and Economics also from Brunel University. He has gained much experience working in industry ranging from distributed systems software development to designing and deploying large-scale computer networks. His research focuses on component-based simulation and issues in distributed simulation interoperability. His email address is <rajeev.sudra@brunel.ac.uk>.

THARUMASEGARAM JANAHAN is a Ph.D. candidate in the Department of Information Systems and Computing at Brunel University, UK having completed a B.Sc. there. His research has focused on Distributed Interactive Simulation and Parallel and Distributed Simulation. He has previous industrial experience in Defense, Banking and Engineering. His email address is <t.janahan@brunel.ac.uk>.

GARY TAN received his M.Sc and Ph.D from the University of Manchester, UK and is now a Senior Lecturer at the School of Computing, National University of Singapore. His research interests are Scheduling and Load Balancing, and Parallel and Distributed Simulation. He has just finished a six-month Commonwealth Fellowship at Brunel University. His email address is <gtan@comp.nus.edu.sg>.

JOHN LADBROOK has worked for Ford Motor Company since 1968 where his current position is Simulation Technical Specialist. In 1998 after 4 years research into modelling breakdowns he gained an M.Phil (Eng.) with the University of Birmingham. In his time at Ford, he has served his apprenticeship, worked in Thames Foundry Quality Control before training to be an Industrial Engineer. Since 1982 he has used and promoted the use of Discrete Event Simulation. In this role he has been responsible for sponsoring many projects with various universities. For the past five years, he has been Chairman of the Witness Automotive Special Interest Group. His email address is <jladbroo@ford.com>.