Investigating Distributed Simulation at The Ford Motor Company

Simon J.E. Taylor
Leif Bøhli
Centre for Applied Simulation Modeling
Brunel University
Uxbridge, Middlesex, UB8 3PH, UK
E-mail: Simon.Taylor@brunel.ac.uk
Leif.Bohli@brunel.ac.uk

Xiaoguang Wang
Stephen J. Turner
Parallel and Distributed Computing Centre
Nanyang Technological University
Nanyang Avenue, 639798 Singapore
E-mail: ASSJTurner@ntu.edu.sg,
xgwang@pmail.ntu.edu.sg

John Ladbrook
Dunton Engineering Centre
Ford Motor Company
Mail 15/4a-F04-D
Basildon, Essex, SS15 6EE, UK
Email:ladbroo@ford.com

Abstract

Engine production is a complex process that requires the manufacturing and assembly of a wide variety of components to create a varied product mix. Simulation plays a key role in the planning process of a new production line to determine if it can meet expected demand. However, these simulations can be very time consuming and can often take up to a day to execute a single run. This paper investigates how distributed simulation based on the IEEE 1516 High Level Architecture and the emerging standard COTS Simulation Package Interoperability Product Development Group (CSPI-PDG) Type I Interoperability Reference Model could be used to reduce the time taken for a single simulation run. CSP interoperability and the problem of integrating CSPs with HLA software (the runtime infrastructure) are presented. New prototype benchmarking software, the COTS Simulation Package Emulator (CSPE), which is being developed to investigate distributed simulation problems, is discussed. The paper then develops a case study of how this was used to investigate the feasibility of using distributed simulation at Ford. The paper discusses results obtained from this case study and suggests that distributed simulation could indeed be beneficial to Ford.

1. Introduction

The production of an engine is a complex process involving the manufacture and assembly of a wide variety of components to create a range of different possible engine types (different capacities, fuel injection options, petrol/diesel, etc.). The requirement for different engines is determined by expected customer demand. When planning a new engine production line to meet this expected demand, many complex factors such as machine cost and reliability, partially built engine test, repair and recycle time, and varying operator shift patterns and availability must be taken into account. In this area, discrete-event simulation is used as the main decision support technique. The planning process is therefore a repeating cycle of production layout creation, model building and simulation, and reporting to determine if new lines can meet expected demand. The COTS simulation package (CSP) WITNESS [14] is used to support modelling and simulation in this process.

Figure 1 shows a typical simulation process used in the design of an engine production line. The aim of this process is typically to determine the sensitivity of a line to changes in factors as outlined above. As can be seen, the process begins when a new production line layout becomes available. A new model is then built and process data is obtained. Validation is then performed on model output and, as the layout cannot be modified, the process data is scrutinized against expected production capacity. Experiments are then run on the model. The ramifications of results from these experiments are then considered and this is repeated until no further experimentation is required.

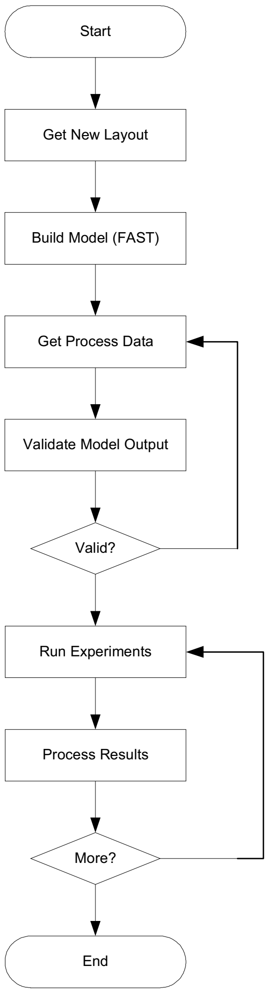

Figure 1. Engine production line simulation process

As part of an on-going collaboration between the Centre for Applied Simulation Modeling at Brunel University (UK), the Dunton Engineering Centre at The Ford Motor Company (UK) and the Parallel and Distributed Computing Centre, Nanyang Technological University (Singapore), work is being carried out to reduce the cycle time of the simulation study involved in this process. Opportunities that have been identified include the reduction of the time taken to build models, to perform a run of a single model, to perform experimentation with the model, and to collaborate with the various stakeholders and the simulation team. These are addressed as follows: by a spreadsheet-based front-end (FAST) to WITNESS that, to a certain extent, automates model building; by investigation carried out in this paper; by distributing experimentation over many processors; and by using groupware [8]. All, apart from distributing the run of a single model have met with varying degrees of success.

Building on previous work that investigated the use of distributed simulation at Ford with an alternative production line layout [11] and contemporary work of the Simulation Interoperability Standards Organization (SISO) CSP Interoperability Product (standards) Development Group CSPI-PDG [9], this paper investigates the feasibility of using distributed simulation techniques based on the IEEE 1516 High Level Architecture and the CSPI-PDG Type I Interoperability Reference Model to reduce the time taken for a single simulation run. WITNESS, like many CSPs, does not have interoperability functionality. This is entirely reasonable as the need for distributed simulation in this area has recently emerged. As this does not currently exist, it is impossible to demonstrate the possible benefits of distributed simulation to a stakeholder. Without this demonstration, the stakeholder cannot express demand for this functionality to the CSP vendor. To solve this, this paper also discusses the development of a CSP *Emulator* (CSPE) that allows us to perform feasibility studies and to demonstrate to stakeholders the possible benefits of adopting this technology. This work is of particular interest to our collaboration and, we hope, the wider simulation community as the execution of a engine production simulation can take over a day to run. Given that experimentation usually requires many runs of a simulation, any possible speedup will therefore represent a significant reduction in the cycle time of a simulation study and, possibly, make it possible for additional experimentation to take place that would not be otherwise possible.

Our paper is structured as follows. CSP interoperability and the problem of integrating CSPs with HLA software (the runtime infrastructure) are introduced in section 2. Sections 3 and 4 outline our approach and the prototype benchmarking software CSPE that is being developed to investigate distributed simulation problems as presented in this introduction. Our case study of how this was used to investigate the feasibility of distributed simulation at Ford is presented in section 5. Section 6 discusses the results obtained from our study and the implications for Ford (and other similar stakeholders). Section 7 concludes the paper with a summary and a short discussion of further work.

2. CSP Interoperability

Consider a model of a factory built in a single CSP. To simulate the factory model, the CSP will use the resources of the single computer on which it runs. However, consider the possibility of dividing the model so that it runs on two computers. The "split" model would run in two CSPs running on two computers. Why would we do this? We do this in the hope of reducing the time taken to simulate the factory by a factor of two. This proposition is not at all new and is one of the main drivers of the field of distributed simulation [2]. However, it *is* relatively new to users of CSPs and is, with a few limited exceptions, currently not possible. Let us explore why.

Briefly, a *federation* is composed of CSPs/model *federates* that exchange data via a runtime infrastructure (RTI) in a time synchronized manner as specified by the IEEE 1516 High Level Architecture (figure 2). Two factories, F1 and F2, generically interact as denoted by the black double-headed arrow. Each model consists of an arrival source So*i*, a queue Q*i*, a workstation W*i*, a resource R*i*, and an exit sink Si*i* (where *i* is the factory identifier). Different types of information might be exchanged. For example, entities might be passed between models (i.e. the two factories are linked together – entities leave F1 at Si1 and arrive in F2 at So2) and the resources R1 and R2 might be shared to reflect a shared set of machinists that can operate workstations W1 and W2. If this was the case, factory F1 must publish and send information to the RTI in an agreed format and time synchronized manner and factory F2 must subscribe to and receive that information in the same agreed certain format and time synchronized manner, i.e. both federates must agree on a common representation of data and both must use the RTI in a similar way. Further, the "passing" of entities and the sharing of resources require different distributed simulation protocols.

Why is this then not possible? Firstly, there is the issue of how a CSP can be integrated with the HLA RTI and secondly, there is the issue of an agreed format and communication protocol. We now consider each issue.

2.1 CSP/RTI integration

Straβburger [7] analyzed the requirements for CSP integration from the perspective of being part of a distributed simulation and the perspective of the programming paradigm. The distributed simulation perspective requires that the CSP should provide an interface at least to connect to an RTI that is capable of exchanging data in a commonly agreed format via a commonly agreed time synchronization protocol. The programming paradigm requires that in addition to this, special consideration must be given to the ambassador paradigm of the HLA [4]; the CSP must be able to communicate with the HLA RTI via the RTIAmbassador and implement the corresponding FederateAmbassador. The approach taken to the programming paradigm can be classified as either *explicit* or *implicit* from the viewpoint of the modeler. The explicit approach requires that the modeler must add HLA synchronization and communication to the CSP/model and the implicit, or *transparent*, approach requires that all this functionality is hidden from the modeler (the functionality is built into the CSP and/or interfacing software between the CSP and RTI).

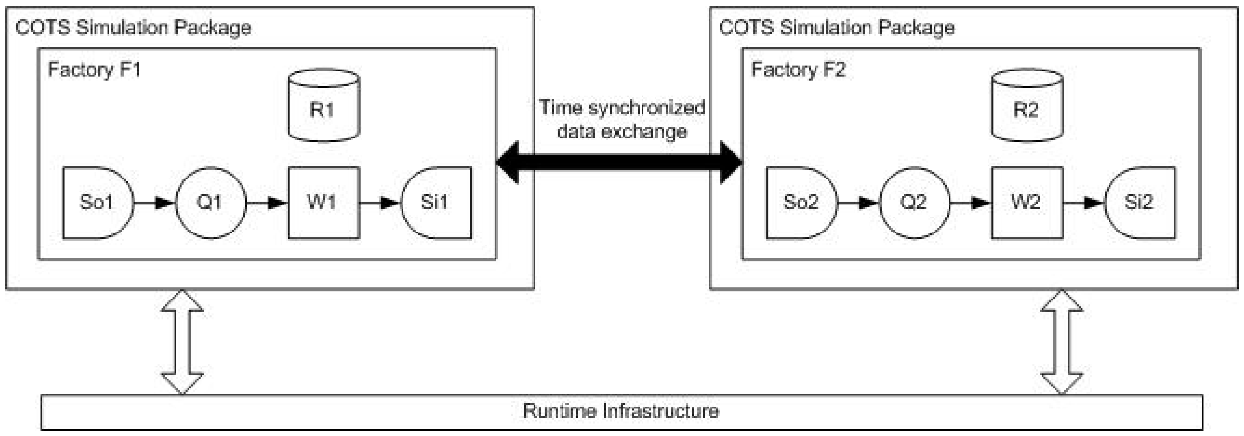

Figure 2. COTS simulation package interoperability

However, as modelers wish to benefit from any possible speedup due to distributed simulation without the cost burden of the extra skills and training required by the explicit approach, the implicit approach is more appropriate in the work described in this paper. However, to achieve this, a CSP needs new features to enable a model built using the package to join a distributed simulation. Some work has been done in this area to suggest new features to be added to CSPs in general to provide transparent interoperability functionality [6].

2.2 Interoperability standards

In addition to the transparency requirements of the implicit approach, there is the problem of agreeing a common standard data exchange format and protocol, i.e. an agreed interoperability standard. In 2004, the CSPI-PDG (Commercial-Off-The-Shelf Simulation Package Interoperability Product Development Group) was approved by the IEEE affiliated Simulation Interoperability Standards Organization (SISO) [9]. Previously known as the HLA CSPI Forum, the Group is dedicated to creating a standardized approach to support the interoperation of discrete event models created in CSPs using the IEEE 1516 High Level Architecture. One of the first issues identified by the CSPI-PDG was the problem of developing a data exchange format and protocol that satisfied all interoperability requirements of distributed simulation with CSPs. The solution to this was the development of a set of Interoperability Reference Models (IRMs) [9]. Each IRM is intended to "capture" a particular interoperability requirement. Currently, these are:

• Type I:	Asynchronous Entity Passing
• Type II:	Synchronous Entity Passing (Bounded Buffer)
• Type III:	Shared Resources
• Type IV:	Shared Events
• Type V:	Shared Data Structures
• Type VI:	Shared Conveyor

Each is intended to be supported by an Interoperability Framework (IF) and a data exchange specification. For example, the Type I IRM *Asynchronous Entity Passing* deals with the common requirement of transferring entities between simulation models. In the Type II IRM *Synchronous Entity Passing*, the input model can transfer entities only when it makes sure that the destination side is not blocked (workstation) or not full (queue) in the receiving model. A solution (an IF) for Type I is intuitively simpler than one for Type II. The IRMs therefore allow progress to be made towards a general solution while providing "intermediate" well-formed solutions on the way. Further, a data exchange specification for *entities* is under development that supports the representation needs of both Type I & II IRMs (other specifications will be created to support the other IRMs) [9]. It is intended that the IF solutions to each IRM will be complementary but capable of operating independently.

In addition to the standards outlined above, a set of COTS Simulation Package Emulators (CSPEs) is being created. As described above, there are currently no CSPs with an efficient interface capable of supporting the IRMs. This is unsurprising as the IRMs are still undergoing product development. However, as redevelopment of CSPs is costly, and progress towards the IFs must be made to create standardized CSP interoperability, a benchmark is needed to compare different possible solutions. This is the purpose of the CSPEs.

This section has outlined some of the problems of CSP interoperability and approaches being taken to create a standardized solution. We will now describe progress made towards the Type I IF on the basis of the Type I IRM.

3. Integrating a CSP with the HLA: Type I IRM

The Type I IRM (Asynchronous Entity Passing) represents models that interact on the basis of entities; models are linked together so that one model may "pass" an entity to another at a given timestamp. The reason why this is termed "asynchronous" is that there is no *immediate or direct feedback* when an entity is passed (this does not mean to say that no feedback can exist, just that it must happen at a different time to when an entity is passed).

In terms of minimum technological support of the logical link between the two models, all that is required is the transmission of timestamped entity information between one model and another in such a way that the receiving model receives the timestamped entity information in the correct order with its own events. This is the reason why this IRM has been termed "asynchronous", there is no synchronous message exchange needed to transfer the entity information between the two models (as is required in the Type II IRM). An IF solution to this Type I IRM must therefore be able to:

- transfer timestamped entity information from one model to another via a timestamped message ,
- allow a model to correctly receive timestamped entity messages from one or more models, and

- correctly coordinate this information with the receiving model events being processed by the COTS simulation package.

In the development of a Type I IF to support this, we have proposed a generic architecture for our interoperability frameworks with the incorporation of a DSManager library and extended RTI [12]. The DSManager provides a generic interface consisting of a set of functions to be invoked by the CSP or CSPE (see Section 4) when necessary. The C++ / Java-based RTI is wrapped by "normal" C functions, that can easily be integrated with most of the current CSPs written in C, C++, Java or VB. The DSManager also interacts with the extended RTI which is developed using a middleware approach [3]. In this extended RTI, known as RTI+, appropriate synchronization algorithms are designed in order to improve the simulation performance and relieve the user from the burden of time management. Examples are a shared state manager [3] for conservative synchronization and a rollback controller [13] for optimistic synchronization.

Our architecture also supports other functions related to distributed simulation, such as the setting of appropriate lookahead values. For example, for benchmarking purposes, a lookahead value can be set and the synchronization approach to be used can be declared. We now introduce our CSPE benchmark software for the Type I IRM.

4. A Type I CSPE

As previously discussed, the range of CSPE benchmarks are intended to facilitate the study of CSP interoperability. The CSPE described in this paper in intended, for now at least, for Type I IRMs. It builds on a previous incarnation described in [10] and represents an important stage in the evolution of the CSPE benchmarks. The main differences are increased functionality and a more flexible user interface for building representative models. This means that this version of the benchmark can investigate a wider range of problems than previously possible.

In our version of CSPE, as with many CSPs, an entity passes through a variety of simulation objects. There are four basic types of simulation objects: entry point, queue (bin or storage), workstation (machine) and exit point. The modeler can define the attributes and property of each entity type, and assign and link necessary simulation objects to process each entity. Entities can be specified either as being "internal" (generated and consumed with the model), or as arriving from another model and/or leaving to another model. If the latter of these are chosen, the name of the source model (arriving from…) and/or destination model (leaving to…) must be specified. Additionally, entry and exits points to and from the model must be specified. Note that in this version of CSPE, for each external entry point, only one type of entity will arrive from only one source model. The modeler can use other external entry points if there are entities from other source models or more than one type of entity from the same model. For exit points, we recognize that if more than one entity leaves a model at the same time then, as with CSP routing, some tie-breaking rule is needed to schedule the ordering of these simultaneous events. In CSPE, the approach used is to assign a different priority to each external exit. The higher the priority (lower value), the earlier (in real time) the entity will be sent out.

A model built using CSPE can be a standalone model or part of a distributed simulation. The modeler needs to choose one and only one of the component models as a controller model. The controller model is in charge of managing the creation and termination of the distributed simulation. The number of component models in the distributed simulation is only needed by the controller model. It is used for the initialization phase of the simulation execution. Each component model also should give the name of the distributed simulation, and the name of the FED Configuration File which is used to supply the RTI with all necessary federation execution details during the creation of a new federation [1]. As discussed in the previous section, in addition to managing the execution of the local model, the CSPE must interact with the DSManager. This includes forwarding necessary information describing the distributed simulation to the DSManager, transferring entities from external exit points, and receiving entities from the DSManager and passing them to the corresponding external entry points. Internally, our CSPE uses the three-phase approach to perform the simulation of the model [5]. Our implementation of this approach has some slight differences as our CSPE uses two event lists (bound event list and conditional event list).

5. Case Study

At Ford our problem is this. Engine production lines are complex but, on the whole, linear. There are some feedback loops, such as when an engine part is inspected for quality and returned back via repair stations into the production line for reprocessing. To study if distributed simulation could benefit such a simulation, i.e. if the simulation processing work could be split between different computers, a case study was developed at Ford. This was based on an analysis of a proposed production line that was being simulated and

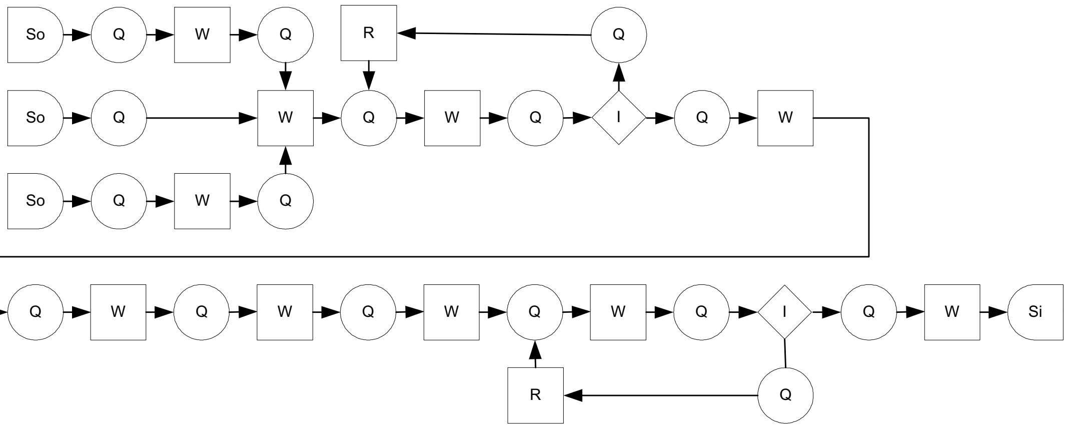

Figure 3. Engine production line model

is shown in figure 3. Actual details of this model cannot be reproduced here due to commercial confidentiality. However, what can be revealed is that it is a "conventional" production line (linear with some feedback loops) and that the queues within the model were sized so that a queue within the model never became full (i.e. the processing times of the machines and the throughput of entities were balanced). Additionally, the numbers and distribution of the queues and the workstations are representative of the real model. The standalone model consists of work stations (squares with "W"), queues (circles with "Q"), repair stations (squares with "R"), inspection stations (diamonds with "I"), entry points ("D"s with directional arrows outward), and an exit point ("D"s with directional arrow inward). Travel time between the different elements in the model was fixed at 10 simulated time units. The time taken to process an entity passing through a work station is set at a fixed number of 14 simulated time units, 1 simulated time unit in the inspection stations and 30 simulated time units in the repair stations. Entities were introduced into the model from the entry points every 14 simulated time units (timings relatively representative of the real system). The three entry points reflect the three classes of engine part that need to be added to the engine assembly as it begins its processing. Overall, the entity passing of the model conformed to the Type I IRM.

Our CSPE could be used to study the feasibility of using distributed simulation to speed up the simulation of this kind of production line. Three different representative scenarios were considered. These are:

- Topology A: Engine production line model with two federates (Figure 4)
- Topology B: Engine production line model with four federates (Figure 5)
- Topology C: Engine production line model with four federates and two repair station federates (Figure 6)

Each of these scenarios was chosen to reflect "real-world" situations where the model might be developed by different modeling teams. Topology A and B represent the division of the model into two and four parts respectively. Topology C represents the further division of the four federate scenario with each of the repair stations placed in separate federates. Note that in the figures additional entry/exit (source/sink) points have been added to the model to reflect arrival and departure points of the entities and that interconnectivity is represented by the appropriate designator (SiA is logically connected to SoA via connector A). The number of federates were limited by license considerations. Many stakeholders have only access to a limited number of licenses of a real CSP and therefore scenarios based on distributed simulation with even six federates represent an investment few companies can actually afford!

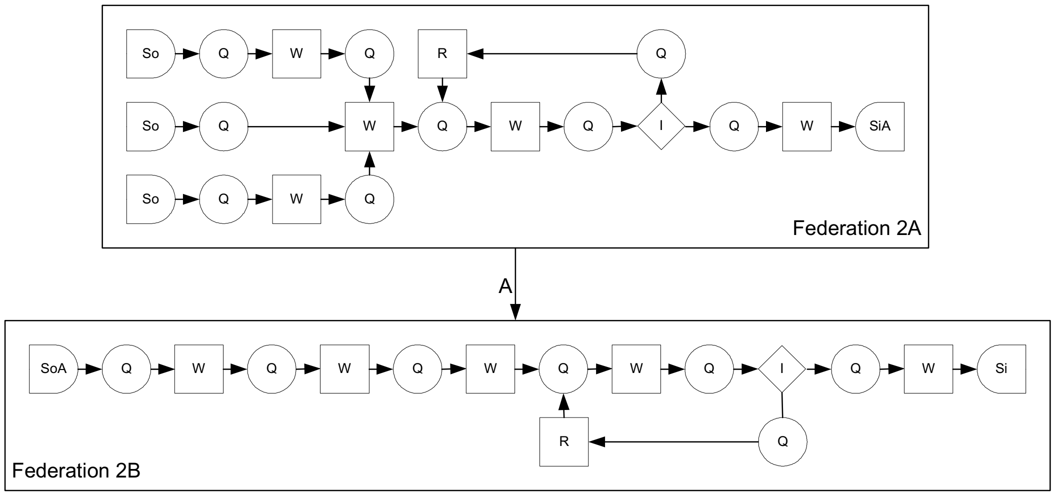

Figure 4. Engine production line model: two federates

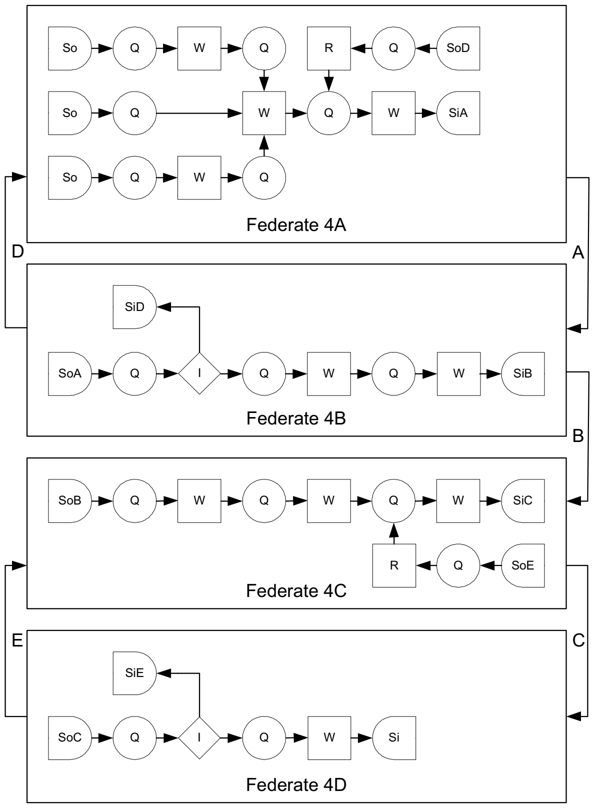

Figure 5. Engine production line model: four federates

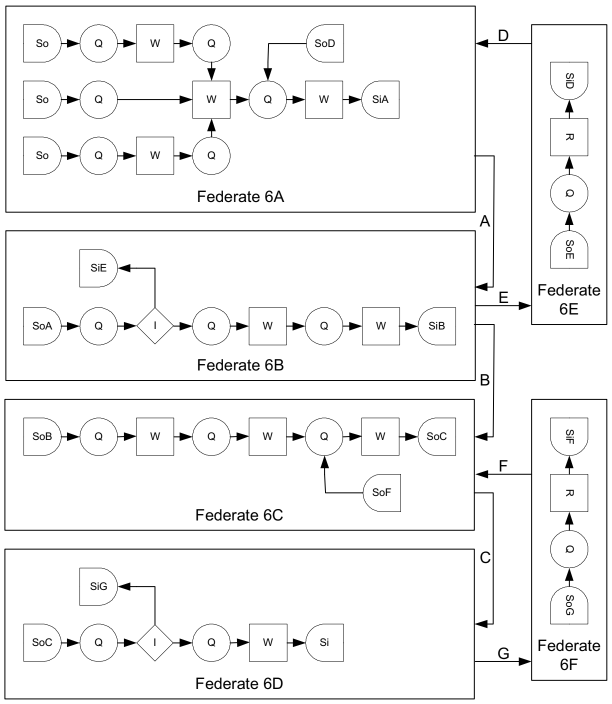

Figure 6. Engine production line model: four federates and two repair station federates

To investigate different processing demands made by the model, something which is entirely possible given a real CSP, we used *event granularity*. We define event granularity as the computation time taken to process an event. In this case study, the event granularity is only used for each workstation to schedule a new event. This allows us to vary computation time to reflect the actions taken during the execution of an event (for example, updating of statistical counters, saves to a trace file, etc.). The event granularity for each experiment was set at 0.001, 0.01, 0.1 and 1 second. Lookahead was fixed relative to the travel time. Parts were rejected and returned earlier in the production line 25% of the time (a representative and not at all realistic figure – the real rejection rate is confidential!). The same number of entities were throughput in all experiments. Our performance tests were carried out on up to six computers connected through an isolated 10 Mbit local area network. The specifications of the machines were Pentium III, 650MHz, 256mb RAM, the router was a Catalyst 2900 10Gb. The RTI Executive was run on a Toshiba Celeron 2.6GHz 450mb RAM.

6. Results

Figure 7 shows how execution time varies with event granularity and figure 8 shows how speedup varies with event granularity (against the runtime of the single model shown in figure 3). As is possibly expected, the difference in execution time is negligible for small event granularities. However, over an event granularity of 0.01, the emergent behavior appears to be that the distributed simulation (any topology) out performs the single model and that execution time reduces with greater numbers of processors. A better demonstration of this is shown in figure 8 as over 0.01, with all topologies speedup increases proportionately as event granularity increases. With event granularities of 0.01, 0.1 and 1.0 respectively, topology A has speedups 1.64, 1.80 and 1.98, topology B has 2.13, 2.37 and 2.53, and topology C has 2.30, 2.58 and 2.95. The trend appears to be that topologies with more processors appear to get better speedup, even though the decomposition is uneven.

What are the implications of this? The representative model of figure 3 at an event granularity of 1.0 takes around 12 hours to execute. This execution time is of a similar magnitude to that of the real simulation run (but in no way represents the number of entities of the real run – the model is more complex!) A simple decomposition into four federates (topology B) yields, with an old RTI, a speedup of around 2.5. Separating out the repair stations (topology C) further increases speedup to around 3. This represents the ability to run the real model in around 4-5 hours – effectively increasing the number of experiments per day to 2-3. It is likely that a more modern RTI could further increase the speed of execution. Given the constraint of limited CSP licenses and limited time to investigate federate decomposition, it appears that this feasibility study indicates a strong reason for Ford to adopt distributed simulation.

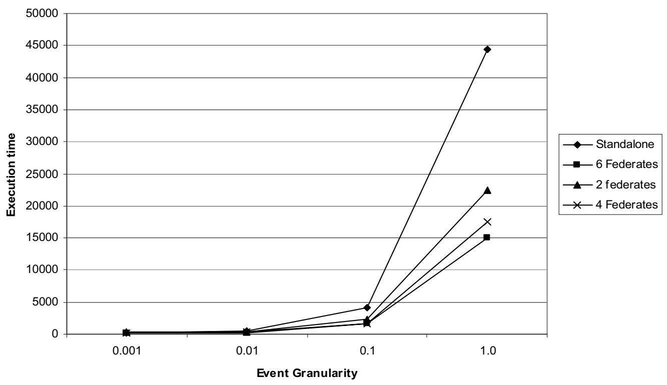

Figure 7. Performance results for engine production line case study (time vs event granularity)

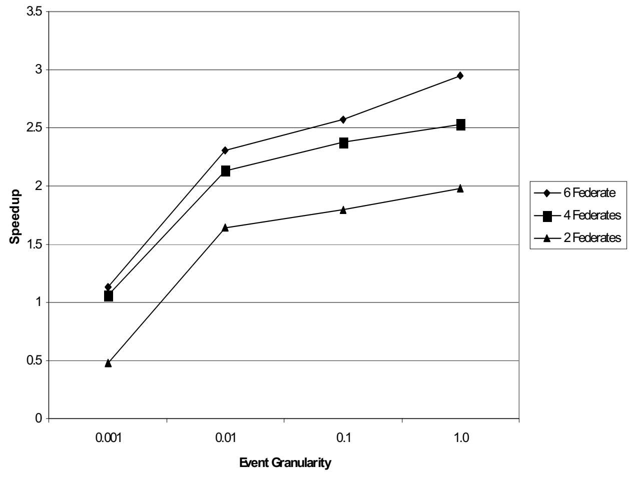

Figure 8. Performance results for engine production line case study (speedup vs event granularity)

7. Conclusions

This paper has investigated the possible benefit of distributed simulation in one area of simulation at The Ford Motor Company. To achieve this, this paper has discussed the CSP interoperability problem and the development of a CSPE based on the CSPI-PDG Type I IRM to investigate these problems. A short case study has been presented and results of experimentation with CSPE show that it may be possible to achieve a speedup in this case that could be beneficial to Ford.

On the basis of this, work is on-going to develop the interface between our DSManager and the WITNESS CSP used by Ford. This will be influenced by success in a sister project to this work in the semiconductor industry as another CSP, Autosched AP [15], has been successful integrated to our DSManager. We hope to further develop our CSPE to support the investigation of Type II IRM problems [13] so that our work can continue to influence the development of distributed simulation solutions and standards in industry.

As a final note, the importance of this work is to show how distributed simulation can potentially benefit real-world applications in an area that is relatively novel such that *stakeholders have confidence* (our validation of CSPE). It is hoped that our contribution of an approach to feasibility studies using our CSPE software will inspire the study of other such applications.

8. Acknowledgements

The development of the CSPE described in this work is part of a pilot project funded by the Agency for Science, Technology and Research (A-STAR) Thematic Strategic Research Program under the theme "Industrial IT - Integrated Manufacturing and Service Systems". The authors would like to thank Thor-Kristian Ingham and Martin Bruusgaard for their help at Ford, as part of the MTech Program run between NITH (Norway) and Brunel, and Malcolm Low (SimTECH, Singapore) in the development of proto standards in this area.

9. References

[1] DoD, *High Level Architecture Federation Execution Data (FED) File Specification – RTI 1.3 Version 3*, Department of Defense, 31 July, 1998.
[2] R.M. Fujimoto, *Parallel and Distributed Simulation Systems*, John Wiley & Sons, New York, 2000.
[3] B.P. Gan, M.Y.H. Low, J.H. Wei, X.G. Wang, S.J. Turner and W.T. Cai, "Synchronization and Management of Shared State in HLA-Based Distributed Simulation", *Proc. 2003 Winter Simulation Conference*, New Orleans, USA, Dec. 7-10, 2003, pp. 847-854.
[4] F. Kuhl, R. Weatherly and J. Dahmann, *Creating Computer Simulation Systems: An Introduction to the High Level Architecture*, Prentice Hall PTR, 1999.
[5] M. Pidd, *Computer Simulation in Management Science*, Wiley, 4th edition, 1998.
[6] M.D. Ryde and S.J.E. Taylor, "Issues Using COTS Simulation Software Packages for the Interoperation of Models", *Proc. 2003 Winter Simulation Conference*, New Orleans, Louisiana. Dec. 7-10, 2003, pp. 772-777.
[7] S. Straβburger, *Distributed Simulation Based on the High Level Architecture in Civilian Application Domains*, PhD Dissertation, University of Magdeburg, Germany, April 2001.
[8] S.J.E. Taylor, S. Robinson and J. Ladbrook, "An Investigation into the Use of Net-Conferencing Groupware in Simulation Modelling", *Journal of Computing and Information Technology*, 13, 1, 2005, pp. 1-10.
[9] S.J.E. Taylor, S.J. Turner and M.Y.H. Low, "The COTS Simulation Interoperability Product Development Group". *Proc. 2005 European Simulation Interoperability Workshop*, Simulation Interoperability Standards Organization, Institute for Simulation and Training, Florida, 2005, 05E-SIW-056.
[10] S.J.E. Taylor, S.J. Turner, N. Mustafee, H. Ahlander and R. Ayani, "COTS Distributed Simulation: A Comparison of CMB and HLA Interoperability Approaches to Type I Interoperability Reference Model Problems", *SIMULATION*. 81, 1, 2005, pp. 33-43.
[11] S.J.E. Taylor, R. Sudra, T. Janahan, G. Tan and J. Ladbrook, "GRIDS-SCF: An Infrastructure for Distributed Supply Chain Simulation", *SIMULATION*, 78, 5, 2002, pp. 312-320.
[12] X.G. Wang, S.J. Turner, S.J.E. Taylor, M.Y.H. Low, B.P. Gan, "A COTS Simulation Package Emulator (CSPE) for Investigating COTS Simulation Package Interoperability", *Proc. 2005 Winter Simulation Conference, to appear*, 2005.
[13] X.G. Wang, S.J. Turner, M.Y.H. Low, B.P. Gan, "Optimistic Synchronization in HLA Based Distributed Simulation", *Proc. 18th Workshop on Parallel and Distributed Simulation*, IEEE Computer Society, 2004, pp. 225-233.
[14] www.lanner.com, viewed 3rd August 2005.
[15] www.brooks.com, viewed 3rd August 2005.