Modelling large scale enterprises: A distributed simulation approach

A.P.R. Wickramarachchi.

A thesis is submitted in partial fulfilment of the requirements of Sheffield Hallam University for the degree of Doctor of Philosophy

May 2004

Abstract

Distributed simulation provides an alternative solution when today's highly complicated systems including manufacturing are to be simulated. Complexities involved in implementation, the need for more expertise, high development cost and long implementation time etc. along with a lack of guidelines for developing distributed simulation, and the complexity of tools and techniques used to implement schemes, resulted in the lack of acceptance for distributed simulation among the general simulation community. In order to address some of these issues, a new approach is proposed for modelling and simulating large scale enterprises using distributed simulation. The proposed approach which includes a comprehensive methodology for distributed enterprise simulation, developed by combining activities required for traditional sequential simulation with additional activities required for distributed simulation.

The thesis elaborates the additional activities required for distributed simulation in different chapters with simplified approaches for carrying out these activities. These include an approach to decide the appropriate simulation strategy (SimSS process), a simplified approach to modelling and model partitioning, a synchronization mechanism that approximately synchronizes the distributed enterprise simulation and an approach for developing distributed simulation using tools and technologies which are popular, well accepted and also cost effective. The differences between the traditional distributed simulation approaches and the proposed methodology include: partitioning of the conceptual model into logical processes before transforming them into computer simulation models, use of commercial simulation software to implement the distributed simulation, and use of cost effective and popular middleware and programming languages. Illustration of the proposed approaches focuses on distributed manufacturing applications.

Acknowledgement

I would like to express my deepest gratitude and appreciation to my director of studies, Dr Sameh M. Saad for his support, guidance and constructive criticism. My appreciation also extended to Professor Terrence Perera for his support and assistance.

My wife Suranga and son Sahan deserve a huge thank you for their support and sacrifices. Last but not least I would like to thank my colleagues, Asian Development Bank and University of Kelaniya, Sri Lanka for the support and the help provided.

Table of contents

Publications made from the thesis

Saad S. M., T. Perera and R. Wickramarachchi. 2002. A new methodology for parallel and distributed simulation, *proceedings of the 2002 International Conference on Responsive Manufacturing.*

Saad S. M., T. Perera and R. Wickramarachchi. 2002. A strategy selection approach in parallel and distributed simulation environment, *International Journal of Advance Manufacturing Systems.* Vol 6, No.2

Saad, S. M., T. Perera, and R. Wickramarachchi. 2003. Simulation of distributed manufacturing enterprises: A new approach. *Proceedings of 2003 Winter Simulation Conference*, 1167-1172

Saad, S. M., T. Perera, and R. Wickramarachchi. 2003, A simplified approach for modelling and simulating large scale enterprises, *European Simulation Symposium*

Saad, S. M., T. Perera, and R. Wickramarachchi. 2004, A simplified approach to develop distributed manufacturing simulations, *2nd International Conference on Manufacturing Research (ICMR 2004)*

Saad, S. M., T. Perera, and R. Wickramarachchi. 2004, A simplified approach for modelling and simulation of large scale enterprises, submitted for *Special Issue of the Transactions of the Society for Modeling and Simulation International*

Chapter 1
Introduction

This chapter, as the first chapter of the thesis provides an introduction to the thesis as well as a rationale for the research carried out. Objectives of the research, contribution of the research and structure of the thesis are also included in this chapter.

1.1 Introduction

Simulation is one of the most powerful tools available to decision makers responsible for the design and operation of many diverse areas of applications. The system under study can be anything from aerodynamic properties of a new aircraft under development to intricacies of protein folding into a complex three dimensional shape. However, the size and complexity of many of today's simulation models place severe demands on the computational resources required. Many researchers have concluded that the traditional sequential simulation has reached its limit in simulating highly complicated and large applications. Distributed simulation (along with parallel simulation) provides an alternative solution when today's highly complex systems are to be simulated by decomposing a simulation model into a number of sub-models and executing them in parallel. Furthermore, a parallel or distributed simulation model is more realistic and natural for many real world applications. Distributed simulation provides a means of executing simulation models scattered over a building, a campus, a city, a country or even the world. It is the only solution available when simulation models need to exist in different places due to various reasons such as security concerns about the confidentiality of information contained in simulation models, simultaneous access to executing simulation models for users in different locations etc. Distributed simulation plays an important role in enterprise simulation. Enterprise simulation refers to a dynamic model or simulation which is constructed with a top-down perspective and is intended to provide an overall view of the workings of an enterprise. In fact, an enterprise simulation model becomes a distributed simulation model when there are two or more simulation models of the enterprise executed in a network. Therefore the term 'distributed simulation' instead of 'enterprise simulation' will be used in the thesis for literature or applications are common to both types of simulations.

This research presents a comprehensive approach for modelling large scale enterprises through distributed simulation. Although the proposed approach focuses on enterprise simulation, it can be employed to develop general distributed simulation models as well.

The objective of this chapter is to provide an introduction to the thesis. The next section explains the rationale for the research. Objectives of the research and contributions made by the research are presented in sections 1.3 and 1.4 respectively. The last section provides the structure of the thesis. The chapter ends with a summary.

1.2 Rationale

Although distributed simulation provides an attractive alternative when simulating large, complex or geographically distributed systems, it is more complex to manage than traditional sequential simulation since inter-processor communication and synchronisation issues required to be addressed. A number of authors have criticised distributed simulation (along with parallel simulation) for the complexity, lack of availability of design and development tools, lack of acceptance by the general simulation community, lack of use in industrial applications etc. Fujimoto (1993a) commented that despite over a decade and a half of research and several successes, this area of simulation has not had a significant impact on the general simulation community. He also predicted that unless new inroads are made in reducing the effort and expertise required to develop (parallel and) distributed simulation models, the field will continue to have limited application, and will remain as a specialised technique used by only a handful of researchers. Although these comments were made almost a decade ago still many authors including Bass (1999), Low et al. (1999), Nicol and Heidelberger (1995), Pham et al. (1998) and Taylor (2002) echo similar sentiments. In spite of a great deal of effort made by the research community, parallel and distributed simulation techniques have not yet been widely used in the industry (Cai and Teo, 1999) and remains a relatively unknown field to those not directly researching in this area (Taylor, 1998). Bass (1999) noted that the complexity and difficulties of implementation have inhibited the commercial success of many parallel and distributed applications. Bagrodia (1996) complained that the design of efficient parallel discrete event simulation models often appears to be a mysterious art primarily practiced by researchers who have been rigorously ordained in this task.

Fujimoto (1993b) concluded that (parallel and) distributed simulation thrives only if the parallel and distributed research community makes the transition process easier for the discrete event community. Turner (1998) noted that, it is generally recognised that the future success of parallel and distributed simulation depends on the extent to which it is possible to reduce the effort and expertise required to develop simulations. This requires simple tools such as user friendly simulation software packages, popular programming languages as well as modelling methodologies that guide parallel and distributed simulation users.

Simulation methodology plays a crucial part in any simulation project, particularly in distributed simulation due to its complex nature. Abrams (1993) highlighted the importance of modelling for parallel and distributed simulation and noted that model design and program design are not independent tasks. Page and Nance (1994) noted that the importance and role of the conceptual framework within the model development process has had little recognition in the parallel and distributed simulation community. Simulation modelling methodologies have been investigated for more than four decades. Different authors have presented a number of methodologies over this period. However, almost all of these were focused on sequential simulation and not on distributed simulation (or parallel simulation either). Therefore it is desirable to investigate methodologies for distributed simulation as this form of simulation is much more complex than traditional sequential simulation. The highly complex nature of this type of simulation calls for more attention into the distributed simulation model development process. Furthermore, less complicated approaches for implementing distributed simulation could play a significant role in addressing some of the previously highlighted issues.

1.3 Objectives of the research

- Present a new methodology for distributed simulation in order to develop large scale enterprise simulation models.
- Simplify the implementation process of distributed enterprise simulation.
- Implement the proposed methodology for distributed simulation using widely available, popular, and cost effective tools and technologies.

1.4 Expected contribution from the research

- A new methodology for distributed enterprise simulation (Chapter 3).
- A process for determining an appropriate simulation strategy (Chapter 4).
- A simplified approach for partitioning enterprise models (Chapter 5).
- An approximate synchronization mechanism for enterprise simulation (Chapter 6).
- An approach to implementation of enterprise simulation models using cost effective, widely used and popular simulation software, middleware and programming language (Chapter 7).

It should be noted that in this thesis the term parallel and distributed simulation will be used instead of distributed simulation as most of the concepts discussed are also relevant to parallel and distributed simulation.

Since different topics are covered in different chapters of the thesis, it was decided not to include a separate literature review chapter. Instead, the background literature relating to different chapters was included within the individual chapters in order to enhance the logical organization of materials presented in the thesis.

1.5 Structure of the thesis

Chapter 1 – Introduction

Provides a rationale for the research carried out, objectives of the research and contributions made by research.

Chapter 2 – Parallel and distributed simulation

Presents an introduction to parallel and distributed simulation, distributed simulation, enterprise simulation, and limitations of parallel and distributed simulation. Discrete event simulation which is the foundation of distributed simulation is also discussed in chapter 2. In addition, it also discusses sequential simulation and limitations of sequential simulation briefly.

Chapter 3 – The new proposed methodology for distributed enterprise simulation

The research carried out for the thesis revolved around the methodology for distributed enterprise simulation. This chapter explains how the proposed methodology developed by combining activities required for sequential simulation with additional activities required for distributed simulation. It presents the literature relating to sequential simulation methodologies, and additional activities needed for distributed simulation before presenting the new methodology. Then it briefly describes stages of the proposed methodology. Subsequent chapters elaborate the key stages of the proposed methodology.

Chapter 4 – The simulation strategy selection (SimSS) process

This chapter provides the user with the SimSS process for determining the most appropriate simulation strategy out of parallel simulation, sequential simulation and distributed simulation. The analytical hierarchy process (AHP) based SimSS process concludes that there is no one best simulation strategy for all situations and the appropriate simulation strategy depends on the situation. Before presenting the SimSS process, the chapter presents literature on the AHP.

Chapter 5 – Conceptual modelling, model partitioning and mapping for distributed enterprise simulations

This chapter briefly discusses literature on modelling methodologies, modelling tools, partitioning and mapping approaches. A simplified approach is proposed for development of the conceptual model for enterprises, partitioning the conceptual model and mapping of the partitioned logical processes.

Chapter 6 – The proposed synchronization mechanism for distributed enterprise simulation

Synchronisation is one of the most important issues to be addressed when developing distributed simulations. The literature relating to synchronisation is presented before proposing an approximate synchronisation approach for distributed simulation. In addition, the literature on networking issues is also discussed briefly in his chapter as distributed enterprise simulation relies on networking infrastructure.

Chapter 7 – Construction of the distributed enterprise simulation model

This chapter describes the approach employed to implement the enterprise simulation using widely available, popular and cost effective technologies and tools. Microsoft Message Queue (MSMQ), Arena and Visual Basic for Applications (VBA) were used as the message passing middleware to connect simulation models, simulation software to build simulation models and VBA to write an interface between MSMQ and Arena respectively. A hypothetical case study is presented to illustrate the implementation process. The chapter also provides a brief discussion of middleware, which is used to communicate between distributed simulation models.

Chapter 8 – Discussion, conclusion and recommendations for further work

The chapter discusses the proposed approaches presented in previous chapters including benefits and shortcomings. Suggestions are also made for improving the work carried out and further work that can be carried out is also included in this chapter.

Summary

This chapter has provided an introduction to the thesis. As noted in the rationale, the research is focused on distributed enterprise simulation methodology, which provides an architecture for the development of enterprise simulation models. The next chapter briefly describes discrete event simulation since distributed simulation is based on the fundamentals of the former and also provides an introduction to parallel and distributed simulation.

Chapter 2
Parallel and distributed simulation

This chapter presents an introduction to the general concepts of simulation and to parallel and distributed simulation. Although the research focused on distributed simulation and in particular distributed enterprise simulation, most of the issues involved are also common to parallel simulation. One notable exception is the location where partitioned model components reside. The literature also uses the term 'parallel and distributed simulation' to refer to distributed simulation. The chapter includes basic concepts in simulation, introductions to sequential simulation, parallel and distributed simulation, distributed simulation, and enterprise simulation which is an application of distributed simulation. A brief discussion of limitations of parallel and distributed simulation which provided a motivation for this research is also presented in this chapter.

2. 1 Introduction

Since the beginning of civilization, people have tried to understand the principles and systems of the environment. An essential tool to this endeavour has been the development of models. In order to model a system, it is necessary to understand the concept of a system and the system constraints. A system is defined as a group of objects that are joined together in some interactions or interdependence toward an accomplishment of some purpose. A system is often affected by changes occurring outside of the system. Such changes are said to occur in the system environment. Depending on the purpose of the study, it has to be decided on the boundary between the modelled system and its environment.

A model is a representation of the construction and working of some systems of interest, and is similar to but simpler than the system it represents (Maria, 1997). A model should be a close approximation to the real system and incorporate most of its salient features. However, it should not be so complex that it is impossible to understand and experiment with it.

A model:

- Acts as a communication vehicle, making available a description of the behaviour of a system.
- Enables users to gain insight and understanding regarding the behaviour of a system.
- Provides means for the analysis and the evaluation of the system as well as the prediction of its future behaviour.

Simulation is defined as an imitation of the operation of a real-world or imaginary process or system over time (Banks, 2000). According to Shannon (1998) simulation is the process of designing a model of a real system and conducting experiments with this model for the purpose of understanding the behaviour of the system and/or evaluating various strategies for the operation of the system. The process of executing a model on a computer system in order to derive answers to questions regarding the operations of modelled systems is referred to as computer simulation. A model adapted for simulation on a computer is known as computer simulation model or simply as a simulation model. The complexity of organized enterprises has enhanced the attractiveness of computer simulation as a problem solving and design tool, and many contemporary systems could only be understood and manipulated using computer modelling and simulation techniques. Shannon (1998) also mentioned that it is one of the most powerful tools available to decision makers responsible for the design and operation of complex processes and systems. Law and Kelton (1991) noted that as a technique, simulation is one of the most widely used in Operations Research and Management Science.

Simulation has a long history and has been used since around 3000 BC, when the Chinese war games *Wei-Hei* were developed. The history of computer based simulation dates back to 1950s (Pidd, 1994) and many fields now rely on extensive use of simulation to test new ideas and options. The gaming industry is exploding with virtual reality and interactive simulations based futuristic games. Interactive simulators have been used for pilot training for years and are increasingly being used for training on advanced equipment. A wide range of simulation applications is available to users in the manufacturing industry. The field of simulation will continue to grow and the technology will move from the domain of more expensive and complex industrial, defence and gaming systems to many aspects of our lives (Jain, 1999). Furthermore, recent technological advances have enabled simulation to be utilized in contexts barely conceivable only a few years ago. Simulation models are now executed not only as conventional 'sequential simulation', but also executed on distributed networks and multiprocessors.

The objective of this chapter is to present an introduction to parallel and distributed simulation, and the foundations of it. The next section describes basic concepts in simulation. Discrete event simulation, on which parallel and distributed simulation is based, is discussed in section 1.3. Section 1.4 presents sequential simulation, limitations of sequential simulation and alternatives to sequential simulation. An introduction to parallel and distributed simulation is provided in section 1.5. The next section describes distributed simulation, and enterprise simulation which is an application of distributed simulation. The last section of the chapter presents a brief discussion of limitations of parallel and distributed simulation. The chapter ends with a summary.

2.2 Basic concepts in simulation

2.2.1 Terminology for simulation

Entity:	Any object or component that requires explicit representation in the model. Entities cause changes in the state of the simulation.
Attributes:	Attributes are characteristics of a given entity which are unique to that entity. They are critical to the understanding of the performance and function of entities in the simulation.
System:	A collection of entities that interact together over time to accomplish one or more goals.
Model:	An abstract representation of a system, usually containing logical and/ or mathematical relationships which describe a system in terms of state, entities and their attributes, sets, events, activities, and delays.
System state:	A collection of variables that contains all the information necessary to describe the system at any time.
Set:	A collection of (permanently or temporarily) associated entities, ordered in some logical fashion.
Event:	An instantaneous occurrence that changes the state of a system.
Resources:	a resource is a type of entity that provides service to other entities.
Activity:	An activity is a definite duration of time that is explicitly defined by the modeller.
Delay:	A delay is an indefinite duration of time that is caused by some combination of system conditions.

These descriptions are based on Banks et al. (1996), Carson (1992) and Ingalls (2002).

2.2.2 Deterministic and stochastic models

If a simulation model does not contain any probabilistic or random components, it is called a deterministic model. In deterministic models, the output is determined once the set of input quantities and relationships in the model have been specified. However, events in most of the real world systems occur randomly. Therefore these systems must be modelled as having at least some random input components. This type of simulation model is known as a stochastic simulation model.

2.2.3 Continuous and discrete models

If the state of the system does not change but remains stable over time (i.e. the system is in an equilibrium state), the system is characterized as a static system. Systems with dynamic behaviour (i.e. systems whose state change over time) are typically classified either as continuous or discrete systems. In continuous systems, state changes occur continuously over time. In discrete systems, state changes are assumed to take place only at a set of discrete instants in time rather than continuously. But in reality, very few real world systems are likely to be entirely discrete or continuous. However, by selecting a certain scope of the respective simulation task one type usually dominates this subsystem (Korn et al., 1999).

2.3 Discrete event simulation

Two types of discrete event simulations emerged that could be distinguished with respect to the way simulation time is progressed (Ferscha and Tripathi, 1994). In time driven discrete event simulation, simulated time is advanced in time steps of constant units. With this type of simulation observation of the simulated dynamic system is 'discretised' by unitary time intervals. Event driven discrete simulation 'discretised' the observations of the simulated system at event occurrence instants. This type of simulation is generally referred to as discrete event simulation.

The behaviour of discrete event dynamic systems can not easily be described by partial differential equations. Several mathematical notations and techniques have been developed to allow the mathematical modelling of discrete event systems including Markov processes, Petri nets, Queuing theory and Finite state mechanics. However, mathematical models are often unable to capture the dynamic behaviour and other important aspects of the system in sufficient detail. Moreover, for most real discrete event systems, mathematical models have no simple and practical, analytical or numerical solutions (Theodoropoulos, 1995). In these situations, simulation modelling and analysis may be the most appropriate alternative for assessing performance of such complex systems (Davis, 1999).

Discrete event simulation has several advantages over mathematical modelling and prototyping (Carson, 1992; Korn, 1999; Shannon, 1992; Shannon, 1998; Thesen and Travis, 1990) including:

- It facilitates the testing and evaluation of systems where the system does not exist. In such cases mathematical modelling is almost impossible, and prototyping is expensive and time consuming. It is generally easier, faster and cheaper to design, build and implement a simulation model.
- Provides higher degree of flexibility than prototypes, as it can be easily modified. Thus it makes possible for efficient experimentation with new situations, providing answers to "what-if" type questions which would otherwise be too time consuming and expensive to contemplate. As a result, it can reduce both system development time and costs.
- Allows the representation of the system at any level of detail sufficient to meet the objectives of the designer by supporting hierarchical design approaches.
- Facilitates the study of dynamic behaviour of systems by allowing the manipulation of time. Time could be compressed or expanded thus providing a rapid view at long time horizons in the past or future of the system under consideration.
- Enhances designers' understanding of the system since the process of discrete event simulation model building requires a detailed description of the system.

Discrete event simulation is based on following building blocks (Pidd, 1994):

- Individual entities

  The behaviour of the model is composed of the behaviour of individual objects of interest, which are usually called entities. The simulation program tracks the behaviour of each of these entities through simulated time. The entities could be truly individual objects such as machines, people, vehicles or could be a group of such objects (i.e. a crowd, a machine shop, a convoy of vehicles).

- Discrete events

  Each entity's behaviour is modelled as a sequence of events, where an event is a point of time at which the entity changes its state. The flow of simulation time in a discrete event simulation is not smooth. As it moves from one event time to another, time intervals may be irregular. A simulation event may be viewed as modelling an event in the physical system, which causes a state transition to take place.
- Stochastic behaviour

  The intervals between events are not always predictable.

2.4 Sequential simulation

In sequential simulation, events are simulated in the order of times at which they occur. Typically a sequential discrete event simulation utilizes three main elements:

- Global clock

  Keep track of the progress of the simulation in terms of logical or simulated time.
- State variables

  Describe the state of the simulation at any particular point in simulated time.
- Event list

  Contains all events which have been scheduled but have not yet occurred.

Each scheduled event is assigned a timestamp, which indicates the point in simulated time at which the event occurs. Simulation of an event may change the event list by scheduling or cancelling pending events. Simulation is carried out by repeatedly removing the next event from the event list, advancing the simulation clock to the time at which the next event is scheduled to occur, and simulating the next event.

2.4.1. Shortcomings of sequential simulation

Simulation of a discrete event system may have a number of objectives to achieve, such as understanding the behaviour of the system, estimating the average performance measures and guiding the selection of design parameters (Righter and Walrand, 1989). However, any simulation tool will be of limited value if it is complicated and difficult to develop and if a simulation takes a very long time to complete. Unfortunately, simulations of large-scale systems are very complicated to develop and take a long time to complete thus greatly restricting the number and scale of experiments that can be performed. Although the speed of sequential processors increases every year, the complexity of the systems also increases every year. Many of today's simulation models place severe demands on the computational resources required (Turner, 1998). Kim et al. (1997) and Fujimoto (1998) noted that simulation of large, complex systems remains a major stumbling block due to its prohibitive computational costs. Complex simulations are slow to develop and slow to run (Carothers, 1999; Righter and Walrand, 1989). The following points summarize problems associated with traditional sequential simulation.

- Complexity of systems

  As mentioned above, simulation models of complex systems are also complex. One solution for complex systems is to model the system at a higher level of abstraction in order to reduce details. Most of the time this is not considered a satisfactory approach as it does not allow the user to incorporate the required details and may end up with an over-simplified version of the system to be investigated. Complex sequential simulation models are difficult to develop and most of the times are unacceptably slow when executing.
- Computational resources

  Generally, complex sequential simulations are slow to run due to their requirements for more resources in terms of more processing power, more memory, and more disk space. If the sequential simulation is executed in a single computer, resources available for the simulation are restricted to the resources available in a single machine. However, for a complex simulation model resources available in a single computer may not be adequate and an obvious means of obtaining a faster simulation is to dedicate more resources to it.
- Parallelism

  Since most simulations are of systems which consist of many components operating in parallel, it could be reasonably assumed of that the inherent parallelism in the system could be exploited, thus improving the efficiency of the simulation. However, sequential simulation models could not exploit this inherent parallelism as they execute events one after another.

Pham et al. (1998) pointed out that the systems we desire to simulate today are so complex that the tasks of executing these simulation models are often beyond the capability of sequential simulators and in many cases sequential simulation has now reached its limits.

2.4.2. Alternatives to sequential simulation

Although the sequential approach to discrete event simulation is based on a very efficient algorithm, it has been unable to provide a satisfactory means of simulating large and complex systems (Calinescu, 1996). In order to overcome this limitation, parallel approaches to discrete event simulation have been considered since the early 1980's by decomposing a simulation for processing on multiple processors. Some of these approaches that were cited in the literature are outlined below (Calinescu, 1996; Hamilton et al., 1997; Koh et al., 1996; Lin, 1993; Righter and Walrand, 1989; Vee and Hsu, 1999).

- Parallelizing compilers

  In this approach, Parallelizing compilers are used to exploit the parallelism available in a given sequential simulation program. It requires no changes in the code for sequential simulation, and thus is readily applicable to many existing sequential simulation programs. However, since the compiler completely ignores the structure of the problem, the parallelism exploited is quite limited. The program may have to be rewritten to exploit more parallelism of the underlying problem.
- Replicated trials (Parallel Independent replicated Simulation – PIRS)

  Under this approach, a number of sequential simulations is run independently on the same number of processors, and their results are averaged in the end. Since no coordination is required among the trials, high efficiency could be expected. However, the parameters of all simulation runs must be decided before any run takes place and this does not encourage interactive decision making. In addition, the computational resources available in computers may impose a severe restriction on the size and complexity of simulations that could be executed this way.

- Distributed functions

  With this approach, the essential subtasks of a simulation are assigned to a number of processors. The subtasks may include random number generation, event set processing, file manipulation, statistics collection etc. This approach requires minimum changes in the code for sequential simulation. However, since the number of such subtasks is limited, not much parallelism could be exploited. Furthermore, the workload among the processors is also difficult to balance.

- Distributed events (with central event list)

  Under this approach, a processor which becomes available continues to process the event with the earliest timestamp in a global event list. The global event list may be maintained either in a distributed manner or by a particular processor. To avoid errors of timing, each processor has to ensure that the event with earliest timestamp in the list will not be cancelled by some events currently processed by other processors. It also has to ensure that processing this event with other events currently being processed by other processors is consistant with the semantics of the system being simulated. This requires knowledge of the simulation model, which may not be extracted easily. The global event list can become a bottleneck if many processors are involved in the simulation.

- Distributed model components

  In this approach, the simulation model is decomposed into loosely coupled components and is assigned the simulation of each component to a process, where one or few processes could be run in a single processor. This decomposition approach is attractive because it is applicable to any model and shows the greatest potential in offering scalable performance for large models, and also for its ability to exploit the inherent parallelism of the simulation model. Since a number of processes runs in parallel, it is required to synchronize the simulation in order to make sure that the simulation progresses correctly.

Among these approaches the distributed model component approach shows the greatest potential and is considered the most promising approach for performing discrete event simulation in parallel (Hamilton et al., 1997; Righter and Walrand, 1989). Since the event list is also decomposed into individual ones, the event list would not become a bottleneck as in the distributed events approach and a higher degree of parallelism is expected since this approach encourages concurrent processing of events with different timestamps (Vee and Hsu, 1999). This approach is generally known as parallel and distributed simulation.

2.5 Parallel and distributed simulation

The field of computer simulation is still growing. As technology develops, old forms of simulations are made faster, and new branches of simulation emerge. This involves taking existing simulation concepts and blending these concepts with those outside of the simulation discipline (Fishwick, 1994). Parallel and distributed simulation combines parallel and distributed computing technologies from computer science with simulation concepts. Pasquini and Rego (1998) pointed that it offers great promise for meeting the simulation needs of developers of increasingly complex systems.

The idea of parallel and distributed simulation was first proposed by K.M. Chandy and R.E. Bryant independently. Papers presented by Chandy and Misra in 1979 and Bryant in 1977 contained basic ideas of parallel simulation, the problem of deadlock and schemes for deadlock resolution, detection and recovery (Overeinder et al., 1991). Jefferson (1985) proposed an alternative scheme for parallel and distributed simulation.

In this thesis the terms parallel simulation and distributed simulation are defined as follows: Parallel discrete event simulation (simply parallel simulation) is concerned with the execution of simulation programs on multiprocessor computing platforms. Distributed simulation is concerned with the execution of simulation on geographically distributed computers interconnected via a Local Area network (LAN) and/or Wide Area Network (WAN) (Fujimoto, 2001). Generally, the term parallel and distributed simulation is used to refer to either or both of parallel simulation and distributed simulations.

Parallel and distributed simulation offers a radically different approach to simulation. In parallel and distributed simulation the system under investigation is partitioned into a number of sub systems. If the system under investigation is a physical system, then the system being simulated is known as the physical system and is considered as a collection of physical processes. In parallel and distributed simulation, physical processes are represented by logical processes (LPs). Hence parallel and distributed simulation could be viewed as a collection of LPs that communicate with each other by passing of timestamped messages. When compared to sequential simulation, it is more complex and requires more expertise of modellers and programmers. Furthermore, a global clock and a global event list do not exist in a parallel or distributed simulation system. However, individual LPs can be considered as sequential simulations with local state variables, a virtual clock and an event list (Mehl, and Hammes, 1993). Bagrodia (1996) also viewed parallel and distributed simulation as a collection of sequential discrete-event simulation models, which communicate with each other using timestamped messages. Since LPs are executed in parallel, their simulation time may advance asynchronously. Thus, a LP may not always receive messages with increasing timestamps. However, in order to simulate a physical system correctly, each LP has to process its incoming messages in its global timestamped order (Cai and Teo, 1999). A synchronized simulation system makes sure that each LP is processing arriving messages in their timestamped order and not in their real time arriving order. This requirement is referred to as the local causality constraint (Fujimoto, 1999). Synchronization mechanisms should allow parallel and distributed simulation to extract maximum possible parallelism and minimize the associated overheads (Sanchez et al., 1996). As synchronization is one of the main issues in parallel and distributed simulation, a more detailed discussion of synchronization will be presented in chapter 6.

Parallel and distributed simulation has attracted a considerable amount of interest in recent years due to large and complex simulations in engineering, computer science, economics, and military applications that consume enormous amounts of time on sequential machines. Lin (2000) noted that since (sequential) simulation is time consuming, it is natural for attempting to use multiple processors to speedup the simulation process. Furthermore, it offers means of exploiting inherent parallelism in real world systems. Dado et al. (1993) also noted that the need for performance, natural concurrency and impact on parallel computation has caused a growing interest in (parallel or) distributed execution of single discrete event simulation. Another potential benefit of utilizing multiple processors is increased tolerance to failures. If one processor fails, it may be possible for other processors to continue the simulation provided that critical elements do not reside on the failed processor (Fujimoto, 1999). Moreover, many researchers agree that a parallel or distributed model is more realistic and natural for many real world applications (Bass, 1999). Davis (1999) mentioned distributed simulation as one area that should provide significant opportunity for further development and application into this millennium. Pidd et al. (1999) predicted that in the future distributed simulation can be evolved into component-based simulation on the web.

As noted earlier, distributed simulations are implemented on workstations connected through a LAN or wide area network WAN. However, LAN based machines have greater communication latencies, although this is gradually being decreased with new networking technologies. WAN based machines experience the highest communication latencies. Communication latency is important, since it is one factor that dictates the efficiency of the simulation system. LAN and WAN based systems often contain computers from different manufacturers. On the other hand, multiprocessor systems are relatively expensive when compared to LAN or WAN based systems. Hence, the use of networked workstations interconnected through LAN/ WAN has been evolving into a popular and effective platform for concurrent execution of simulation models.

2.6 Distributed simulation

Although, parallel computers are much more widely available than was the case a decade ago, they are far from universal and, their use is not straightforward and may require yet more specialist knowledge (Cassel and Pidd, 2001). The proliferation of inexpensive and powerful workstations has continued at a rapid rate in the last few years. In recent years the use of networked workstations for distributed applications is gaining popularity (Ikonen and Porras, 1998). The cost of this method can be kept down, as most of the equipment is already available.

The low cost of equipment and incremental scalability are the other main advantages of using a distributed system. Generally, most of the time workstations are used for small tasks or they are idle and these idle cycles could be utilized to run parallel application on networks of workstations. The network of workstations can be considered as a parallel computer, or 'hypercomputer', whose performance is similar to that of a parallel machine but is achieved at much lower cost (Cabillic and Puaut 1997). Furthermore, network computing environments retain their ability to serve as a general purpose computing platform and run commercially available software products. Fujimoto (1999) presented several reasons that encourage distributing the execution of simulation across multiple computers.

- Reduced execution time

  Execution time could be reduced by subdividing a large simulation into many sub-models that can execute concurrently. This is possible due to availability of more computational resources and exploitation of parallelism inherent to most of the real world systems. However, it must be noted that parallel simulation reduces execution time further with its low communication latencies.
- Geographic distribution

  Executing the simulation program on a set of geographically distributed computers enables creation of a virtual single simulation program of which components are distributed across different physical sites. Allowing the user to keep sub-models of the simulation where they belong may alleviate security concerns about leaking of sensitive information and simplify the sub-model maintenance process.
- Ability to connect computers from different manufacturers.

  Unlike in parallel computers which use processors from the same manufacturer, distributed simulation allows connecting of different computers from different manufacturers. This also helps to keep costs at lower level.

Panda and Ni (1997) noted that since LAN technology was not initially developed for parallel processing the communication overheads among workstations are still quite high. Low communication speeds, shortage of network bandwidth and the ever increasing demand for network resources may result in slowing down the execution speed of the distributed simulation model. Although the networked workstations will be slower than dedicated machines, they may be fast enough and may require much less specialist expertise to put them to use, at a fraction of a cost of the price needed for a dedicated parallel processing computer (Cassel and Pidd, 2001). Thus, investigating the use of distributed simulation using standard networking technologies seems to make much sense.

The research was focused on enterprise simulation. However, as noted in the previous chapter enterprise simulation model becomes a distributed simulation model when simulation models of two or more enterprises are linked together and executed in distributed environment.

2.7 Enterprise simulation

Historically, discrete-event simulation has been view as a standalone project based technology. However, as technology advances at rapid pace, it is envisioned that the next wave of simulation applications may bring simulation to a higher level of applicability in the business application arena. Mastaglio (1999) highlighted simulating business enterprises as the next major application approach to use simulation technology effectively. Enterprise simulation which is considered as an important application of distributed simulation does so by promising to extend the benefits of simulation modelling and analysis as it is performed today. Moreover, advances in distributed simulation concepts and networking technology can provide much needed push to enterprise simulation by serving as enablers. The success of enterprise simulation in simulating war games back in 1990 proved its role in analyzing the behaviour of complex systems, which by definition are comprised of a number of independent systems.

Although the term enterprise simulation or something similar is being used with ever-increasing frequency, the field lacks a clear definition and discussion of the theoretical basis for what is meant by the term. Enterprise simulation can be viewed as a simulation, which is constructed with a top-down perspective and is intended to provide an overall conceptual view of the workings of the enterprise (Mastaglio, 1999; Meilke, 1999). It provides decision-makers a virtual environment in which they can quickly, economically, and safely test and improve their understanding and expertise about the environment. For this purpose, different functions in an enterprise should be identified, modelled and integrated together to run as a single simulation system with distributed simulation technologies. Datar (2000) noted that an enterprise simulation model becomes a distributed simulation model when there are two or more simulation models of the enterprise in the network. If the enterprise consists of more than one organization, the use of geographically distributed enterprise simulation allows each partner to hide any proprietary information in the implementation of the individual simulation. Furthermore, it also allows simulation of multiple systems at different degrees of abstraction level, to link simulation models built using different simulation software, to take advantage of additional computing power, simultaneous access to executing simulation models for users in different locations, to reuse of existing simulation modes with little modifications etc. (Gan et. al., 2000; McLean and Riddick, 2000; Taylor et. al., 2001; Venkateswaran et. al., 2001).

2.8 Limitations of parallel and distributed simulation

Parallel and distributed simulation research began more than 20 years ago as a means of speeding up the execution of discrete event simulation by distributing simulation workload across a number of processors. It offers a great promise for meeting the simulation needs of developers of increasingly complex systems (Pasquini and Rego, 1998). However, the widespread interest in parallel and distributed simulation in the research community did not bring about the widespread deployment of it in real world applications (Fujimoto, 1993). Bagrodia (1996) complained that the design of efficient parallel discrete event simulation models often appears to be a mysterious art practiced primarily by academic researchers who have been rigorously ordained in this task. Presumably more than 1500 research papers have appeared (since the pioneering work by Chandy and Misra, and Jefferson) which have significantly contributed in the scientific sense, but nevertheless failed to bring the field to an industrial and/or commercial success (Ferscha et al., 2001). Ikonen and Porras (1998) noted that distributed simulation is generally not considered as an option when companies are deciding about their simulation methods. Execution of a discrete event program on a parallel computer is no trivial task. Even though the system being simulated often contains much intrinsic parallelism, translating this into concurrent execution of the simulator has proven to be challenging (Fujimoto, 1993). Pancake (1996) also noted that, although parallelism is an intuitive and appealing concept, in practice parallelism carries a high price tag. Parallel programming involves a steep learning curve and is also effort intensive.

Following are some of the reasons cited in the literature which contributed to lack of success of parallel and distributed simulation commercially (Bagrodia, 1996; Ferscha et al., 2001; Fujimoto, 1993).

- The confluence of the parallel and distributed simulation strategy, the execution platform and the simulation model on performance is not properly understood.
- The interweaving of simulation model, platform and strategy attributes and their impact on overall simulation performance is overwhelmingly complex.
- The preference among optimistic and conservative synchronization protocols, given simulation model and platform attributes, is neither conclusive nor can protocol optimizations establish a general rule of superiority.
- The potential performance gain which can be achieved through the use of shared memory multiprocessors, distributed memory multiprocessors or network of workstations is not conclusive. Neither fast processors (on their own) nor fast communication (on its own) can guarantee performance gain.
- The relation of development cost, performance gain and utility of parallel and distributed simulation is not well understood. Moreover, the utility aspect has almost always been excluded from parallel and distributed simulation research work.
- Simulation codes must be developed by computer programmers whose expertise lies in parallel and/or distributed computing and not in simulation modelling, while the simulation model must be developed by simulation practitioners whose expertise lies in simulation modelling and not in computer programming. Although this is common to any form of simulation, distributed simulations are especially affected due to the fact that more expertise is required to implement a distributed simulation

- The focus of much parallel and distributed simulation research remains on the design of a parallel simulation model rather than on the design of a discrete event simulation model for which parallelization can be explored as one execution option.

Parallel and distributed simulation represents a trade-off to the user: the carrot is reduced execution time, but the stick is the effort required to modify, or perhaps even rewrite, the simulation program to effectively exploit concurrency, not to mention the time required to obtain the necessary expertise to accomplish this task (Fujimoto, 1999). Many users avoid parallel and distributed simulation because it is difficult to specify a large and complicated model using existing tools available in this type of simulation. Very few attempts have been made by commercial companies to experiment with parallel and distributed simulation, not to mention deploying simulations in the companies (Low et al., 1999). Furthermore, existing literature on parallel and distributed simulation is justifiably viewed from the outside as having little relevance to industrial simulation (Nicol and Heidelberger, 1995).

Summary

This chapter provided a description on the basic concepts of simulation and, parallel and distributed simulation. It also discussed limitations of sequential simulation and presented alternatives to sequential simulation. An introduction to parallel and distributed simulation, distributed simulation, enterprise simulation and limitations and problems associated with parallel and distributed simulation are also included in the chapter. Key issues involved with parallel and distributed simulation such as synchronization, model partitioning, and networking aspects are presented elsewhere in the thesis in detail. Some of the problems associated with distributed simulation that are mentioned in the last part of this chapter provided a motivation to develop the proposed new methodology for distributed enterprise simulation, which will be presented in the next chapter.

Chapter 3
The proposed methodology for distributed enterprise simulation

Parallel and distributed simulation provides an attractive alternative to sequential simulation when simulating large, complex or geographically distributed systems. However, it was highlighted in the previous chapter that distributed simulation is still not widely used apart from military applications. Absence of a proper methodology to develop distributed simulation, complexity of it, requirements of more expertise etc. were cited in the literature as reasons for lack of popularity among the general simulation community. This chapter proposes a new methodology for distributed enterprise simulation by incorporating additional activities needed for distributed simulation into activities required to carry out a sequential simulation. As noted in a previous chapter, an enterprise simulation becomes a distributed simulation when more than one enterprise simulation models are executed in a distributed manner. Therefore the methodology for distributed enterprise simulation also can be viewed as a methodology for distributed simulation.

3.1 Introduction

Simulation methodology focuses on the question of how a simulation model should be constructed. Balci (1990) noted that the key to success in a simulation study is to follow a comprehensive life cycle which contains key activities to be carried out in an organized and well managed manner. Ever-increasing complexity of systems being simulated can only be managed by following a structured approach to conduct the simulation study. Simulation modelling methodologies or simulation life cycles have been investigated for more than four decades. Different authors have presented a number of methodologies over this period. However, almost all of these were focused on sequential simulation and not on parallel and distributed simulation. Sawhney (2000) pointed out that current simulation modelling methodologies are not geared towards these types of complex systems. Furthermore Karacal (1998) mentioned that modelling and simulation still lacks sound theoretical and methodological foundations. Therefore, the development of efficient discrete event simulation methodologies remains an important area of investigation. Analysis show that the literature on distributed simulation (including parallel simulation) concentrated on a few critical issues such as synchronization, partitioning, mapping etc. However, it fails to mention how systems under investigation are decomposed into logical processes, how distributed systems are verified and validated, the importance of conceptual modelling etc. Absence of formal simulation model building methodologies may have partly contributed to parallel and distributed simulation's failure to gain a significant acceptance from the general simulation community. The task of developing a distributed simulation is especially complex as models are generally larger and more complicated than traditional sequential simulation models, the system under investigation needs to be decomposed into several models, distributed simulation models require to be synchronized, and output can be generated from more than one model of the system. Therefore it is desirable to have a formal modelling methodology which guides users through the different stages required to implement a distributed simulation system in order to simplify and streamline the system development process. The proposed methodology for distributed enterprise simulation was developed by identifying additional activities required for distributed simulation and incorporating them into the key activities required to develop a sequential simulation (figure 3.1)

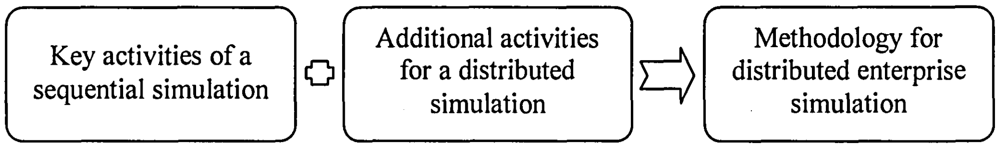

Key activities of a sequential simulation
Additional activities for a distributed simulation
Methodology for distributed enterprise simulation

Figure 3.1 – Approach employed to develop the proposed methodology

The objective of this chapter is to present a new methodology for distributed enterprise simulation. The next section presents the literature on sequential simulation methodologies. Additional activities required for distributed simulation are discussed in section 3.3. The proposed new methodology for distributed enterprise simulation is presented in section 3.4. The last section briefly describes the stages of the proposed methodology. The chapter ends with a summary.

3.2 Sequential simulation methodologies

Methodologies for conducting a simulation have been proposed by a number of authors since early days of the technology. Whilst they do not always share the same terminology, analysis suggests that key activities are common to all. In order to identify the key activities required to conduct a successful simulation, a few well recognized methodologies were selected and analysed.

3.2.1 Methodology proposed by Robinson (1994)

Robinson (1994) presented a simple methodology which consists of 4 main phases (figure 3.2). Both forward and backward arrows on sides have been used to illustrate the iterative nature of the simulation project.

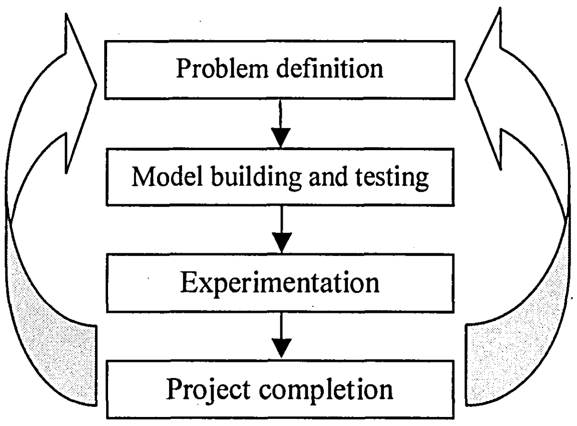


Figure 3.2 – Methodology proposed by Robinson (1994)

The four main phases of the methodology consist of a number of sub activities including,

Problem definition	Problem identification & setting of objectives
	Definition of experimental factors and reports
	Determination of the scope and level of the model
	Collection and analysis of data
	Providing project specifications
Model building and testing	Structuring of the model
	Model building and verification
	Model validation
Experimentation	Performing experiments
	Analysis of results and conclusions
Project completion	Communication of results
	Completing the documentation
	Reviewing the project
	Further work

3.2.2 Key activities of Shannon (1998)

Shannon (1998) listed followings as activities required to complete a simulation successfully.

Problem definition
Project planning
System definition
Conceptual model formulation
Preliminary experimental design
Input data preparation
Model translation
Verification and validation
Final experimental design
Experimentation
Analysis and interpretation
Implementation

3.2.3 Methodology proposed by Balci (1990)

Balci (1990) proposed a 10-phase methodology, which he called the life cycle of a simulation study (figure 3.3).

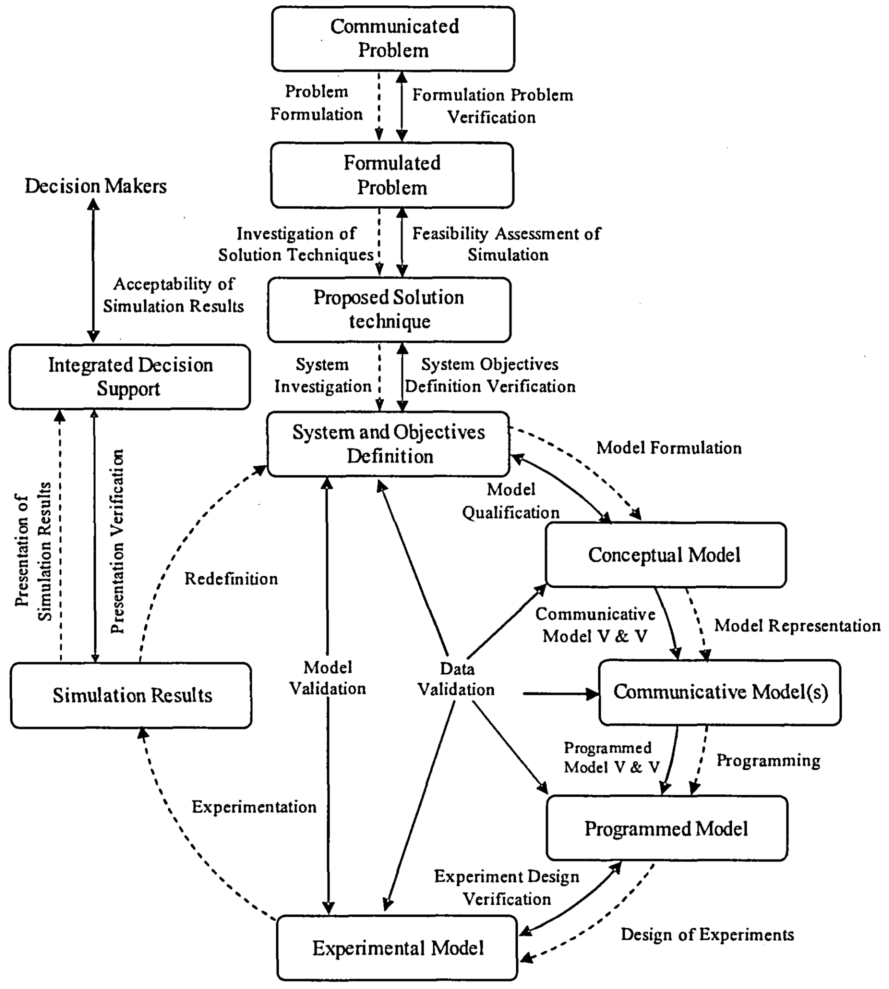

Communicated Problem
Problem Formulation
Formulation Problem Verification
Formulated Problem
Decision Makers
Investigation of Solution Techniques
Feasibility Assessment of Simulation
Acceptability of Simulation Results
Proposed Solution technique
Integrated Decision Support
System Investigation
System Objectives Definition Verification
System and Objectives Definition
Model Formulation
Model Qualification
Presentation of Simulation Results
Presentation Verification
Redefinition
Conceptual Model
Communicative Model V & V
Model Representation
Simulation Results
Model Validation
Data Validation
Communicative Model(s)
Programmed Model V & V
Programming
Experimentation
Programmed Model
Experiment Design Verification
Design of Experiments
Experimental Model

Figure 3.3 – Methodology proposed by Balci (1990)

A key feature of this approach is that most of the activities are centred on verification and validation. In addition to activities involved in the simulation project, Balci (1990) also included the outcome of each activity in the methodology. Key activities of the methodology are shown below.

Problem formulation

Investigation of alternative solution techniques

System investigation

Model formulation

Model representation

Programming

Design of experiments

Experimentation

Redefinition (of model)

Verification and Validation are applied to all phases

3.2.4 Methodology proposed by Law and Kelton (1991)

A 10 step simulation methodology proposed by Law and Kelton (1991) is shown in figure 3.4.

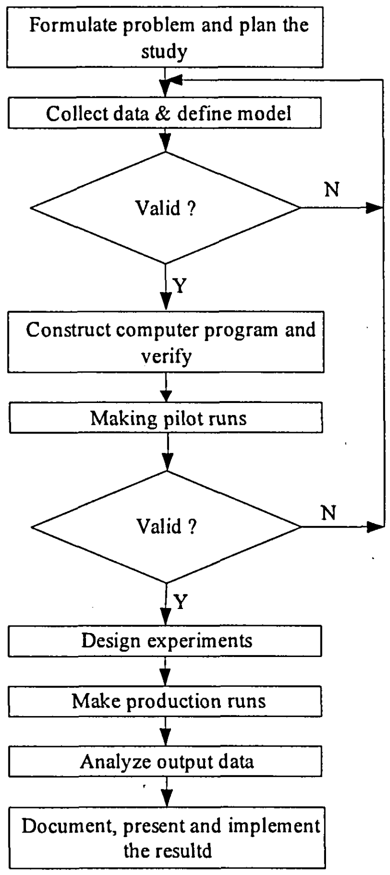

Formulate problem and plan the study
Collect data & define model
Valid ?
N
Y
Construct computer program and verify
Making pilot runs
Valid ?
N
Y
Design experiments
Make production runs
Analyze output data
Document, present and implement the resultd

Figure 3.4 – Methodology proposed by Law and Kelton (1991)

Key activities of this methodology include:

Problem formulation and planning of the simulation study

Data collection and definition of the conceptual model

Validation of conceptual model

Construction and verification of computer simulation program

Making pilot runs

Validation

Design of experiments

Making production runs

Analyzing output data

Documentation, presentation and implementation of the results

3.2.5 Methodology proposed by Banks et al. (2000)

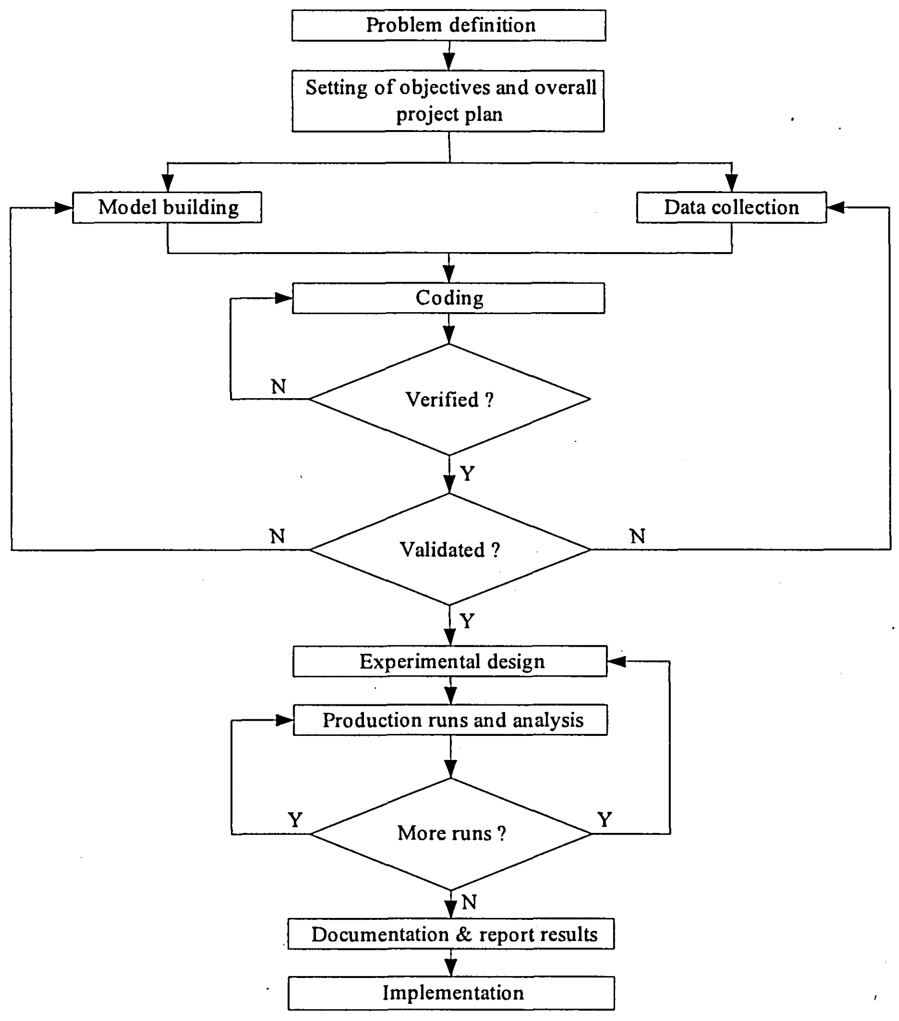


Figure 3.5 – Methodology proposed by Banks et al. (2000)

Methodology proposed by Banks et al. (2000) consists of following steps.

Problem formulation

Setting of objectives and overall project plan

Data collection

Model building

Coding

Verification

Validation

Experimental design

Production runs and analysis

Need for more runs?

Documentation and reporting of results

Implementation

In addition, a number of authors including Lilegdon (1996), Maria (1997), Nordgren (1995), Sadoun (2000), Sargent (1994) also presented activities required for successful sequential simulation.

While some of the above methodologies are concise with few activities (Robinson, 1994), others are lengthy (Bank et al, 2000; Law and Kelton, 1991). The latter methodologies elaborate the activities of the former into a number of activities. Analysis of simulation methodologies suggests that sequence of activities to be carried out are also not in the same order although the key activities required to be carried out are almost the same. A number of authors including Law and Kelton (1991) and Robinson (1994) suggested that data collection should be carried out after problem identification and setting of objectives, and before formulation of the conceptual model. However, Shannon (1998) suggested carrying out data collection after completing problem definition, project planning, system definition, conceptual model formulation and preliminary experimental design. Banks et al. (2000) placed model building and data collection at the same level arguing requirements of data depends on requirements of model building. Some methodologies (Banks et al., 2000; Shannon, 1998) proposed conducting verification and validation at latter part of the methodology after construction of the computer simulation model but before experimentation.

On the other hand, Law and Kelton (1991) proposed verification and validation to be done after completion of key activities such as formulation of the conceptual model and conversion of the conceptual model into a computer simulation model. Balci (1990) proposed to conduct verification and validation throughout the model development process.

Having analyzed above simulation methodologies, the following activities were identified as key activities required to conduct a simulation.

- Problem definition and identification of objectives
- Data collection
- Construction of conceptual model
- Verification and validation
- Construction of computer simulation model
- Experimentation
- Output analysis
- Implementation and further work

Problem identification and identification of objectives was selected as the first stage of the distributed simulation. This stage includes problem definition (Robinson, 1994; Shannon, 1998), problem formulation (Balci, 1990; Banks et al., 2000), project planning, system definition (Shannon, 1998), setting of objectives (Banks et al., 2000). Data collection was selected as the next activity assuming that in order to construct the conceptual model data has to be collected beforehand. Preliminary experimental design (Shannon, 1998) is also included into this stage as it contains identification of input data, statistical distributions that represent data etc. Construction of the simulation model is presented in all the above mentioned sequential simulation methodologies as a separate activity except in the methodology presented by Law and Kelton (1991), where data collection and defining a conceptual model were incorporated into a single stage. However, Balci (1990) presented this stage as two separate stages, namely: model formulation and model representation. Verification and validation were proposed to be carried out at different levels after completing important stages of the simulation project such as completion of the conceptual model, partitioning of the conceptual model into logical processes and transformation of logical processes into computer simulation models. Design of experiments (Balci, 1990; Law and Kelton, 1991), experimental design (Banks et al., 2000), final experimental design (Shannon, 1998), experimentation (Balci, 1990; Shannon, 1998), production runs (Banks et al., 2000; Law and Kelton, 1991), additional runs (Banks et al., 2000) were included in the experimentation stage of the proposed methodology. Output analysis stage includes simulation results (Balci, 1990), analysis of output data (Law and Kelton, 1991), analysis and interpretation (Shannon, 1998), and results analysis (Banks et al., 2000). Implementation and further work consists of project completion (Robinson, 1994), redefinition of the model (Balci, 1990); document, present and implement the results (Banks et al., 2000; Law and Kelton, 1990), and implementation (Shannon, 1998).

3.3 Additional activities for parallel and distributed simulation

The major difference between sequential simulation and distributed simulation is the number of processors (in workstations) used to execute the simulation. Sequential simulation executes as a single model in a single processor. In distributed simulation the entire model is partitioned into logical processes and executed in more than one workstation in a distributed environment. For the purpose of executing a simulation as a distributed simulation, in addition to the key activities mentioned above, the following activities are also required to be carried out. These activities were identified by analysing the literature on parallel and distributed simulation.

- Partitioning of the entire model into logical processes
- Deciding on synchronization protocols and networking aspects
- Assigning of logical processes to different processors

Although distributed simulation has a great potential to improve discrete event simulation, it doesn't provide a simple or standard solution for complex simulations. Many authors including Ikonen and Porras (1998) and Pancake (1996) complained that distributed simulations (including parallel simulations) are effort intensive, complex and costly. Simulationists need to be aware of the benefits of distributed simulation as well as its perils and pitfalls before making a decision on whether or not to use distributed simulation. Therefore it was decided to incorporate an additional activity which guides users to identify an appropriate simulation strategy out of sequential simulation, parallel simulation or distributed simulation.

Figure 3.6 shows activities required for a distributed simulation by combining above mentioned additional activities into activities required for sequential simulation.

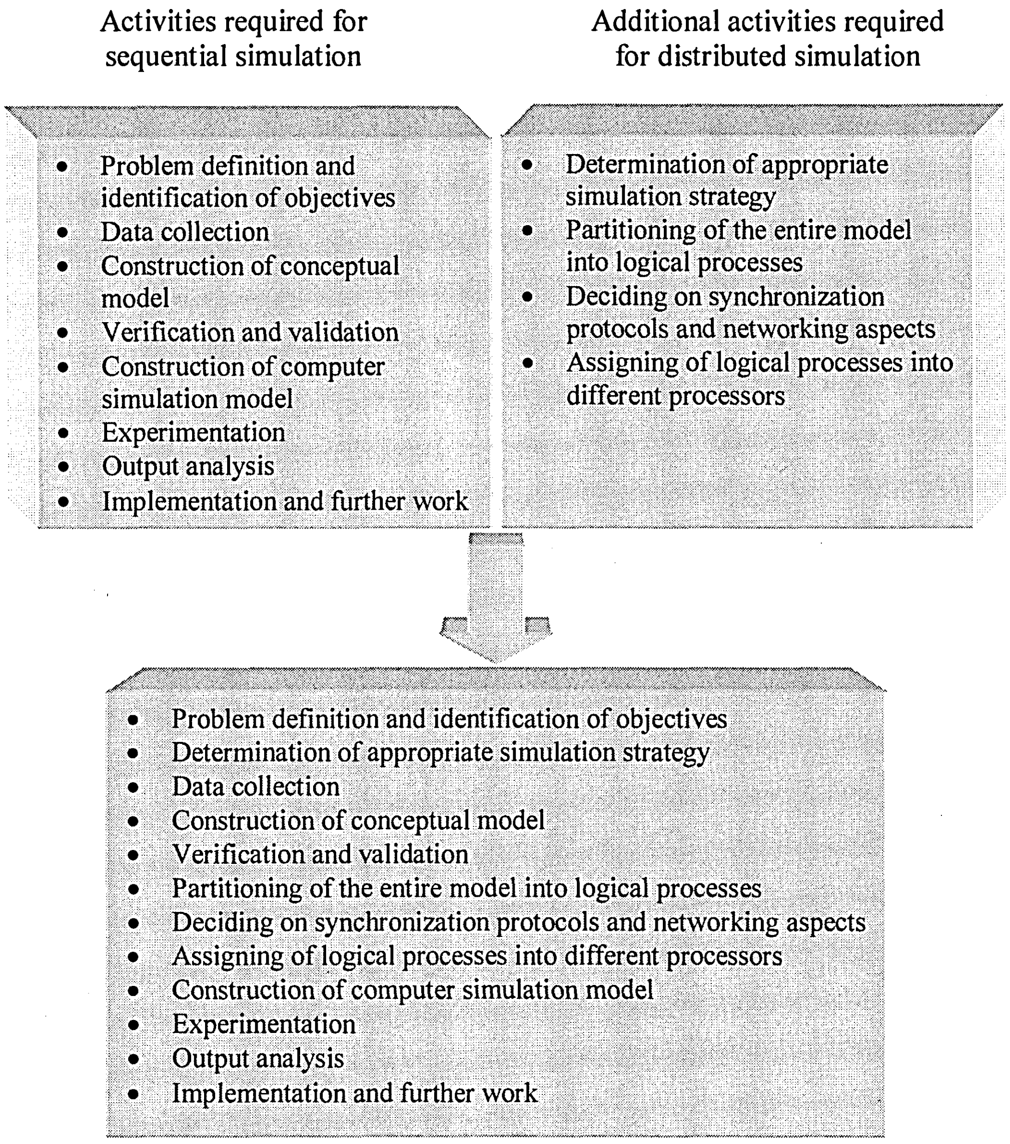

Activities required for sequential simulation
Problem definition and identification of objectives
Data collection
Construction of conceptual model
Verification and validation
Construction of computer simulation model
Experimentation
Output analysis
Implementation and further work
Additional activities required for distributed simulation
Determination of appropriate simulation strategy
Partitioning of the entire model into logical processes
Deciding on synchronization protocols and networking aspects
Assigning of logical processes into different processors
Problem definition and identification of objectives
Determination of appropriate simulation strategy
Data collection
Construction of conceptual model
Verification and validation
Partitioning of the entire model into logical processes
Deciding on synchronization protocols and networking aspects
Assigning of logical processes into different processors
Construction of computer simulation model
Experimentation
Output analysis
Implementation and further work

Figure 3.6 – Activities required for a distributed simulation including distributed enterprise simulation

Based on the activities identified in figure 3.6, the new proposed methodology for distributed enterprise simulation is presented in the next section.

3.4 The proposed methodology for distributed enterprise simulation

The proposed methodology is shown in figure 3.7 and stages of the methodology are described in the following sections. Key additional activities required for a distributed simulation with proposed implementation approaches are presented in the next three chapters of the thesis.

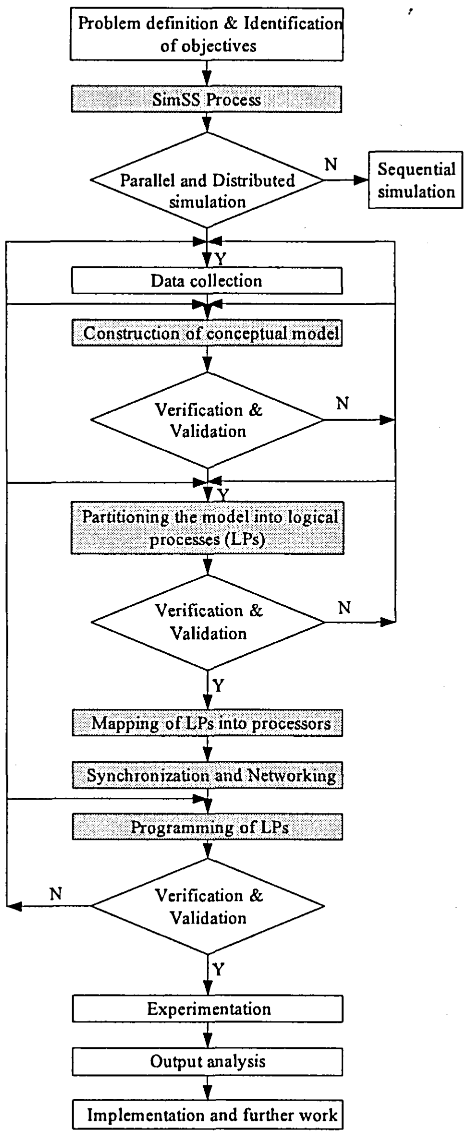

Problem definition & Identification of objectives
SimSS Process
Parallel and Distributed simulation
N
Sequential simulation
Y
Data collection
Construction of conceptual model
Verification & Validation
N
Y
Partitioning the model into logical processes (LPs)
Verification & Validation
N
Y
Mapping of LPs into processors
Synchronization and Networking
Programming of LPs
Verification & Validation
N
Y
Experimentation
Output analysis
Implementation and further work

Figure 3.7 - The proposed methodology for distributed enterprise simulation

New approaches are presented for highlighted stages of the proposed methodology (figure 3.7) in order to implement the distributed enterprise simulation in simplified and cost effective manner.

3.4.1 Problem definition and identification of objectives

As with any project, without proper understanding of the problem and a clear set of objectives it is almost impossible for a simulation effort to succeed. This is particularly true for distributed simulation (and parallel simulation), since it is more complex than conventional sequential simulation. Shannon (1998) noted that beginning a simulation project properly may make a critical difference between success and failure. Specific questions to be answered by the simulation project, systems configurations to be modelled, performance measures used to evaluate different system configurations, and the time frame of the project including cost and resources required are to be determined at this stage. In addition, the scope of the project and abstraction level of the model has to be decided too. The scope of the model is vital to success of the simulation effort as too little detail may result in information that may not be accurate enough to achieve the real goal and, a model with too much detail requires more effort to create, needs longer run times and is more likely to contain errors. Additional resources required for distributed simulation such as expertise, computer networks, and special software if required also need to be considered at this stage. A number of authors including Robinson (1994) and Sadowski (1991) provided an in-depth discussion of the starting phase of a simulation project. Although these discussions were originally produced for sequential simulation, they are also applicable to distributed simulation.

In distributed simulation, more than one model can generate output. In some situations part of the output from a model may need to be restricted to only owners of that model. Therefore it is particularly important to determine which models generate output, which part of the output can be accessed by all the interested parties and which part of the output needs to be restricted.

3.4.2 The Simulation strategy selection (SimSS) process

The second step of the proposed methodology, the SimSS process helps users to determine the most appropriate simulation strategy to be used when executing a simulation model. After analyzing advantages and disadvantages of sequential simulation and parallel and distributed simulation, it presents three alternative strategies, namely: sequential simulation, distributed simulation and parallel simulation. To identify the appropriate simulation strategy, the SimSS process employs the analytical hierarchical process (AHP). The SimSS process will be presented in detail with illustrations in chapter 4 of the thesis.

3.4.3 Data collection

At this stage data is to be collected in order to specify model parameters and probability distributions. Data is required not only to build the simulation model but also to test its validity. Soundness of the model logic and structure depends upon data on which the model is going to be constructed. Amount of data and accuracy of data required also depend on the experimental requirements of the simulation. Therefore before starting the actual data collection effort, experimental design aspects such as measures of effectiveness to be used in the study, what factors going to be varied, how many levels of each of these factors will be investigated and the number of samples need for the study have to be taken into account. In addition, consideration should be also given to type of data required, availability of data, whether data is pertinent and valid, and how to collect the data. Moreover, statistical sampling, statistical distributions, random number generations are also playing critical role in data collection and preparation for a simulation study. More details on input modelling, sampling, data collection and statistical distributions are presented by Law and Kelton (1991), Leemis (2001), Robinson (1994) and Wilson (1997).

3.4.4 Construction of the conceptual model

The conceptual model that represents the real world or proposed model is a series of mathematical and logical relationships concerning the components and the structure of the system under investigation. A conceptual model is a collection of information that describes a simulation developer's concept about the simulation and its pieces. That information consists of assumptions, algorithms, characteristics, relationships, and data, which describe how the simulation developer understands what is to be represented by the simulation (entities, actions, tasks, processes, interactions etc.) and how that representation will satisfy simulation requirements (Pace 2000). Proper development of the conceptual model is vital as it is the primary mechanism for transforming simulation requirements into specifications that can guide the simulation development and implementation process. Conceptual modelling is largely software independent and particularly important for distributed simulation due to its complicated nature. Unfortunately, the literature on distributed simulation has not paid as much attention to conceptual modelling as it deserves. An appropriate modelling approach and technique must be determined before developing the conceptual model. Modelling approaches (such as incremental and hierarchical approaches) specify the way models are to be developed. Once the approach is decided then the modeller can determine what modelling tools (such as diagrammatic tools, Petri nets, and IDEF methodologies) are to be used. Further discussion of this stage along with partitioning of the conceptual model and mapping of logical processes is presented in chapter 5.

3.4.5 Verification and validation

Verification and validation is an important and well researched area in simulation as accuracy and reliability of outcome of the simulation depends on proper functioning of the model as well as validity of the model and data used. Sargent (2000) described verification as ensuring that the computer program of the computerized model and its implementations are correct. Validation is determination that the conceptual model is an accurate representation of the system under investigation. It is often too expensive and time consuming to determine that the model is absolutely valid for its purpose. Instead, tests and evaluations can be conducted until sufficient confidence is obtained that a model can be considered valid for its intended application (Sargent, 2001). Furthermore, Carson (2002) noted that validation is not absolute and any model is a representation of the system, and its behaviour is at best an approximation to the system's behaviour.

According to the proposed methodology for distributed enterprise simulation, verification and validation are carried out at multiple stages, namely: after construction of the conceptual model, after partitioning of the conceptual model into logical processes, and after converting logical processes into computer simulation models (figure 3.7). If errors or omissions are discovered, which is almost always the case then the conceptual model or logical processes must be modified before proceeding into the next stage. Once logical processes are transformed into computer simulation models, individual simulation models are to be verified to ensure that they are working without any bugs and validated to make sure that they produce intended output. Validation of sub models may not always feasible as they are designed to use parameters from other models as input. However, this problem can be overcome by initially designing sub models to generate their own input parameters and able to run as independent models, then modifying them to receive parameters from other models once they are validated. After individual simulation models are verified and validated, the distributed simulation model can be validated in conjunction with synchronization mechanism (more details are presented in section 7.9.2). Analysis of the simulation literature shows that a number of authors including Balci (1990 and 1998), Carson (2002), Law and McComas (2001) and Sargent (2001) presented an excellent introduction to verification and validation including verification and validation procedures, tools and techniques that can be used for verification and validation.

3.4.6 Partitioning and mapping

At this stage, the validated conceptual model is partitioned into several logical processes and assigned to processors (of workstations). The first process is generally known as partitioning and the latter is known as mapping. The issue of model partitioning and mapping has been paid less attention in the parallel and distributed simulation literature compared to the amount of work devoted to other issues such as synchronization (Solcany et al., 1995). The literature on partitioning suggests a number of approaches to partition a programmed simulation model (Boukerche and Trooper, 1994; Nandy and Loucks, 1993). Some of these approaches require execution of the whole simulation system as a single model before partitioning. However, with this new methodology it is proposed to partition the conceptual model before transforming into a computer program. The proposed approach simplifies the conversion of logical processes into a computer program, and verification and validation of the system. It also facilitates the involvement of more than one modeller and computer programmer. When partitioning, interactions within logical processes and between logical processes should be taken into account in order to minimize the inter-processor communication, which reduces the load of the network. Once the system under investigation is partitioned into logical processes, they must be validated before being mapped into to different processors. More details of this stage along with construction of the conceptual model is presented in chapter 5.

3.4.7 Synchronization protocols and networking issues

A Synchronized simulation system makes sure that each individual simulation model is processed arriving messages in their timestamped order and not in real time arriving order. This requirement is referred to as local causality constraint (Fujimoto, 1999). To satisfy the local causality constraint, a number of synchronization protocols has been proposed. These protocols can be broadly classified as conservative or optimistic protocols (Fujimoto, 1990). Conservative approaches strictly impose the local causality constraint and guarantee that each model will only process events in non-decreasing timestamp order. In contrast, optimistic approaches allow violations of local causality constraint, but are able to detect and recover by rolling back to the point where the violation has occurred and reprocessing events in timestamped order. Neither conservative nor optimistic classes of synchronization algorithms proved to be strictly better than the other (Das, 2000 and Sanchez et al., 1996). The appropriate synchronization protocol should be selected based on characteristics of the model and user requirements. (Ferscha et al., 2001; Fujimoto, 1998). In addition, for distributed simulation appropriate network topologies and communication protocols should also be determined. Chapter 6 presents more details on networking issues and synchronization protocols.

3.4.8 Construction of computer simulation model

At this stage validated logical processes are transformed into computer simulation models. Although simulation software packages such as Automod, Promodel, Arena, Witness are used in sequential simulation, general purpose programming languages such as C++, Pascal etc. are often used to develop distributed simulation models. This is mainly due to lack of support offered by simulation software packages for special requirements of distributed simulation such as synchronization. However, analysis of the recent literature (Hibino et al., 2002; Lendermann et al., 2001; McLean and Shao, 2001; Taylor et al., 2001; Venkateswaran et al., 2001) shows increasing trend of using commercial simulation software along with programming languages and message-passing technologies to develop distributed simulation systems. Ability to use commercial simulation software packages may also popularize the use of distributed simulation in industrial applications. Chapter 6 discuses the construction of a distributed enterprise simulation model in detail with an illustration using a hypothetical case study.

3.4.9 Experimentation

All the work carried out in previous stages will not be fruitful if simulation experiments are not carefully planned and designed. Barton (2001) also noted that simulation projects could fall short of their intended goals unless the simulation model is exercised intelligently to gain a better understanding of the likely performance of the system under investigation. Experimentation stage includes both designing of simulation experiments and actual execution of the simulation distributed system. According to Antony (1998) experimental design is a systematic and structured approach to experimentation. When designing simulation experiments, various issues need to be considered including warm-up period, number of replications and length of a replication. To improve the confidence in the estimate of system performance obtained through simulation experiment, Sherif (1998) suggested that longer runs and more replications need to be carried out. Attention should also be paid on starting conditions of the simulation, selection of samples, sample sizes and ways of collecting output. Barton (2001) presented a five step procedure to carry out simulation experiments. Experimental design is also well researched area in simulation and a number of authors including Centeno and Reyes (1998), Kelton (2000), and Wild and Pignatiello (1991) presented more details on this area.

In addition, when conducting a distributed simulation starting and stopping of the simulation may needs a careful consideration as more than one simulation model have to be started and stopped instead of a single model in sequential simulation. Since the new methodology for distributed simulation proposes to use general purpose networked workstations to run distributed simulation models, timing of the simulation experiment may also play an important role. If simulation is carried out at a time when network traffic is very high, it may affect performance of the simulation as well as overall performance of the network.

3.4.10 Output analysis

Output analysis is used to estimate measures of performance for the scenarios that are being simulated. The purpose of analyzing results is to check the extent to which the objectives of the simulation project have been achieved. Various techniques such as graphical analysis, tabular forms can be used to organize the output from the simulation experiment. If complex and detailed analysis is required, output can be written to a text file or exported to a database or spreadsheet package. Since the input processes driving a simulation are usually random variables, generally the output generated from the simulation is also in random nature. Goldsman and Tokol (2000) noted that raw output data is not independent, not identically distributed and also not normally distributed. This leads to difficulties of applying statistical techniques to analyze the simulation output. Output analysis techniques depend on whether the simulation is terminating or nonterminating. Goldsman and Tokol (2000), Nakayama (2002), and Sanchez (2001) presented output analysis techniques in detail.

Unlike in traditional sequential simulation model where output is generated by only one model, in a distributed simulation more than one model can generate output. Output from individual models can be used to measure performance of different sections of the enterprise and the aggregated output can be used to measure the overall performance of the enterprise.

3.4.11 Implementation and further work

No simulation study can be considered as successful unless its results have been understood, accepted and implemented. The last stage of the distributed simulation methodology includes communication of results, documentation, review of the project and deciding on further simulation experiments. To overcome potential resistance against organizational changes, results of the simulation, and benefits of implementation should be clearly, concisely and convincingly documented and presented before implementation starts. Furthermore, use of animation may also help to convince about working of distributed simulation to sceptical audience.

Summary

This chapter presented the proposed new methodology for distributed enterprise simulation, and briefly described key stages of the methodology. Stages which are part of sequential simulation methodologies were described only briefly as these are well researched areas. Additional activities required for distributed simulation will be presented in subsequent chapters. The next chapter presents the SimSS process, which helps users to determine an appropriate simulation strategy out of sequential simulation, parallel simulation and distributed simulation.

Chapter 4
The Simulation Strategy Selection (SimSS) process

The previous chapter presented the proposed methodology for distributed enterprise simulation. It was also noted that, although (parallel and) distributed simulation provides an attractive alternative for sequential simulation, it is more complex and requires more effort and cost to implement. Therefore, it is desirable to evaluate different simulation strategies before making a decision on which simulation strategy is to be selected. This chapter presents the simulation strategy selection (SimSS) process to determine the appropriate simulation strategy out of parallel simulation, distributed simulation and sequential simulation (figure 4.1). It also describes the analytic hierarchy process (AHP), which was used as the solution method for the proposed SimSS process.

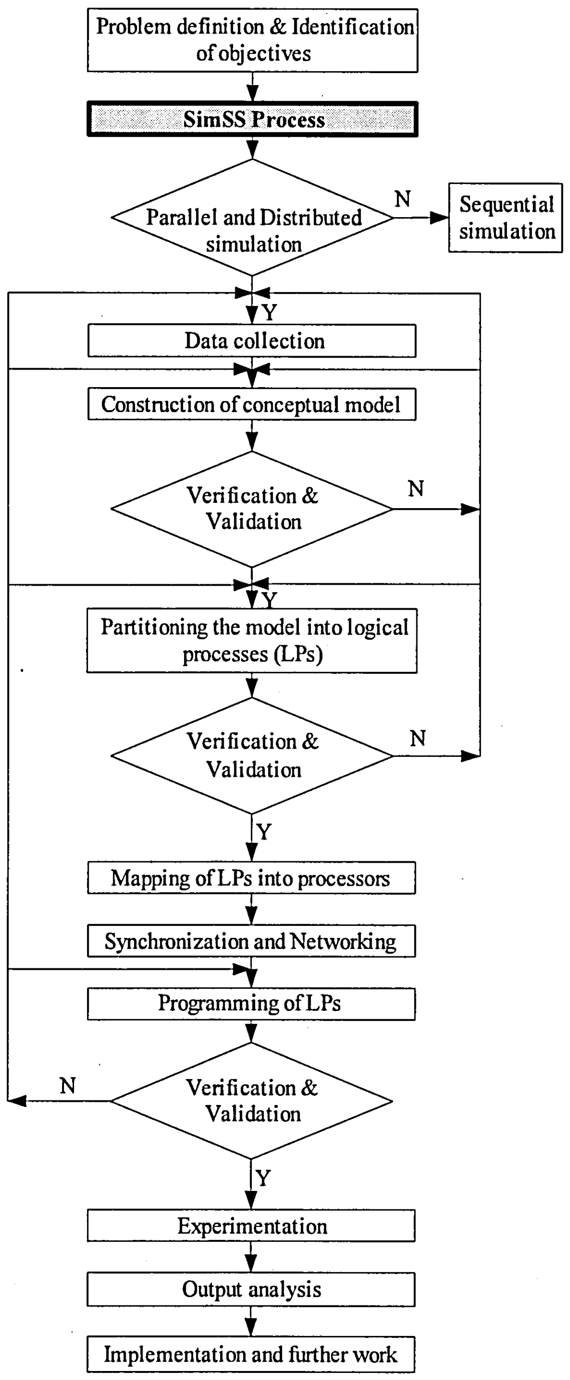

Problem definition & Identification of objectives
SimSS Process
Parallel and Distributed simulation
N
Sequential simulation
Y
Data collection
Construction of conceptual model
Verification & Validation
N
Y
Partitioning the model into logical processes (LPs)
Verification & Validation
N
Y
Mapping of LPs into processors
Synchronization and Networking
Programming of LPs
N
Verification & Validation
Y
Experimentation
Output analysis
Implementation and further work

Figure 4.1 - The proposed methodology for distributed enterprise simulation

4.1 Introduction

Distributed simulation (along with parallel simulation) has a great potential to improve discrete event simulation. Davis (1999) noted distributed simulation as one area that should provide significant opportunity for further development and application into this millennium. Research in this area began more than 20 years ago as a means of improving simulation execution time. Yet, it doesn't provide a simple or standard solution for complex simulations. Parallel programming is also effort intensive and involves steep learning curves (Pancake, 1996). When compared to sequential simulation, parallel and distributed simulation is more complex and requires more expertise of modellers and programmers.

Analysis of the literature suggests that parallel or distributed simulation is employed when speed of simulation needs to be increased by exploiting the inherent parallelism of the system under investigation, more computational resources are required for simulation and/ or simulation needs to be executed in a geographically distributed environment etc. However, if the model is required to be run in a geographically distributed environment then distributed simulation is the only available option for simulation users. Factors that encourage users to move into distributed simulation can not be quantified and decisions made are often subjective. Generally these decisions depend on the expertise of the modeller, availability of resources including both computational and human, enthusiasm of the management and/ or modellers etc. One modeller may decide to use distributed simulation for a particular situation while another modeller may decide to stick with sequential simulation. Thus, there is a need for a systematic approach to select an appropriate simulation strategy by identifying and prioritizing relevant criteria, and evaluating the trade-offs between technical, economic and performance aspects.

It was noted in the previous paragraph that the decision making process involved when selecting distributed simulation (or parallel simulation) is a multi-criterion and judgmental one. The role played by each factor varies from one situation to another. For an example if the simulation model is relatively small, execution time is not critical, and the simulation model can be developed as a single model then sequential simulation might be the appropriate strategy. On the other hand, if the model is too complicated to develop as a single model and execution time is longer than expected then parallel or distributed simulation might be an appropriate alternative. Therefore, it is not feasible to present a definite set of rules to make such decisions. It is further complicated by attributes that are subjective and not quantifiable. Multi-criteria decision Making (MCDM) techniques are useful in circumstances that necessitate the consideration of different courses of action, which cannot be evaluated by the measurement of a simple or single dimension.

Many authors including El-Mikawi (1996), Poyhonen and Hamalainen (2001), Steuer and Na (2003) and Zanakis et al. (1998) describe a number of MCDM techniques including goal programming, outranking approaches, direct point allocation (DIRECT), simple multi-attribute rating technique (SMART), swing weighting, trade-off weighting, multiple objective programming, multi-attribute utility analysis, multicriteria decision analysis and analytic hierarchy process (AHP). Based on simplicity and easy to use, availability of software, and capabilities of software available; the AHP was selected as the MCDM technique for simulation strategy selection. AHP provides a framework to cope with multiple criteria situations involving intuitive, rational, qualitative and quantitative aspects (Chan et al., 2001). Since simulation strategy selection involves multiple criteria, most of which are qualitative and subjective, AHP is an appropriate technique for the proposed simulation strategy selection process (SimSS).

The objective of this chapter is to present a new approach to select an appropriate simulation strategy from sequential simulation, parallel simulation or distributed simulation. The next section describes the AHP, which was used as the solution process for the SimSS approach. Section 4.3 presents the SimSS process and three scenarios to illustrate the proposed process. The chapter ends with a summary.

4.2 The analytic hierarchy process (AHP)

The analytic hierarchy process (AHP), developed by Thomas Saaty in 1970s allows decision-makers to model a complex problem in a hierarchical structure showing the relationships of the goal, objectives (criteria), sub-objectives, and alternatives (figure 4.2). It enables decision-makers to derive ratio scale priorities or weights as opposed to arbitrarily assigning them. In doing so, the AHP not only supports the decision-makers by enabling them to structure the complexity and exercise the judgement, but also allows them to incorporate both objective and subjective considerations in the decision process (Forman, 2001). Yusuff et al., 2001 commented that the AHP provides remarkable versatility and power in structuring and analyzing the complex multi-attribute decision problems. The AHP has been widely used as a decision making tool in many diverse areas including software evaluation, information systems outsourcing, reliability evaluation of distributed computing environments, advanced manufacturing systems, project management, competitive bidding processes, and vendor selection (Al-Harbi, 2001; Cagno et al., 2001; Chan et al., 2001; Fahmy, 2001; Ossadnik and Lange, 1999; Tam and Tummala, 2001; Yang and Huang, 2000; Yusuff et al., 2001). Al-Habri (2001), Perez (1995) and Zahedi (1986) discussed shortcoming and benefits of the AHP.

In AHP, the goal is a statement of the overall objective. The AHP criteria used as basis for the decision are known as objectives. Objectives can be further elaborated into sub-objectives if necessary. Pair wise comparisons of elements (usually alternatives and criteria) can be established using a scale (Table 4.1) indicating the strength with which one element dominates another with respect to a higher level element. This scaling process can then be translated into the priority weights (scores) for comparison of alternatives. Yusuff et al. (2001) presented the following steps of the AHP solution process based on Saaty's work.

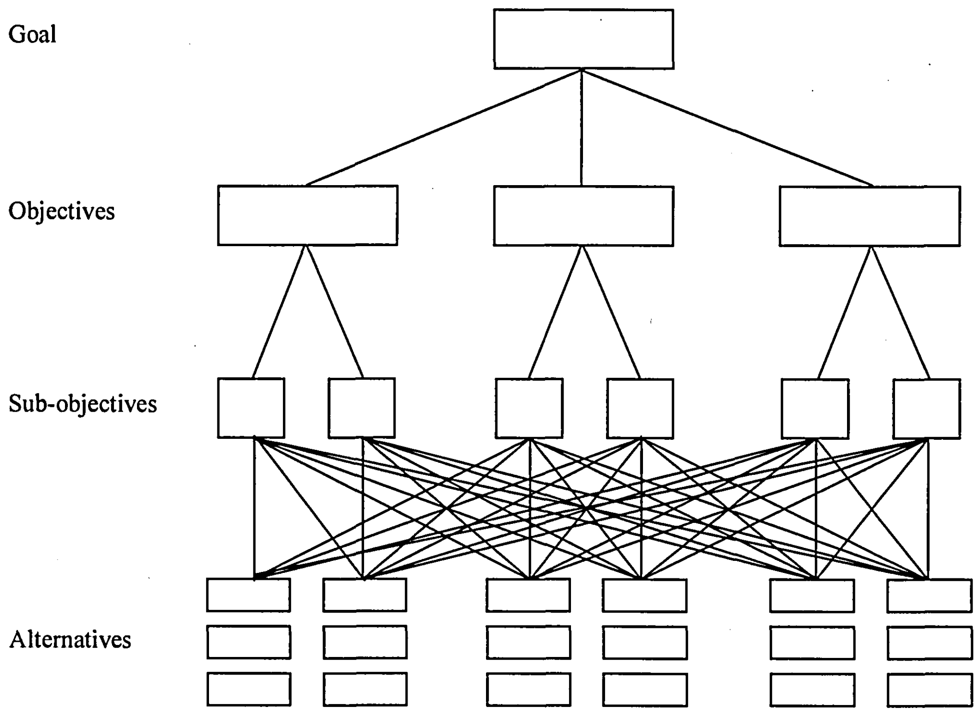


Figure 4.2-AHP decision hierarchy

- Determination of the relative importance of the attributes (objectives) and the sub-attributes (sub-objectives), if any

- Determination of the relative standing (weight) of each alternative with respect to the sub-objective, if applicable, and then successively with respect to each objective.

- Determination of the overall priority weight (score) of each alternative.

- Determination of the consistency indicator(s) in making pair wise comparisons. This step is optional and AHP provides a measure of inconsistency in each set of judgements. However, Forman (2001) noted that real world problems are hardly consistent.

Intensity of importance	Definition	Explanation
1	Equal importance	Two activities contribute equally to the objective
3	Weak importance of one over another	Experience and judgement slightly favour one activity over another
5	Essential or importance	Experience and judgement strongly favour one activity over another
7	Very strong or demonstrated importance	An activity is favoured very strongly over another; its dominance demonstrated in practice
9	Absolute importance	The evidence favouring one activity over another is of the highest possible order of affirmation
2, 4, 6, 8	Intermediate values between adjacent scale values	When compromise is needed
Reciprocals of above nonzero	If activity *i* has one of the nonzero values assigned to it when compared with activity *j*, then *j* has the reciprocal value when compared with *i*	A reasonable assumption

Table 4.1 – AHP scale and meaning

In order to simplify the decision making process it was decided to use the AHP based software that enables users to calculate the priority levels without manual calculations. Ossadnik and Lange (1999) evaluated AHP based software namely: AutoMan, Expert Choice and HIPRE 3+ (using AHP), and concluded that Expert Choice received the highest priority among the three AHP based software. The criteria selected for this evaluation include graphical presentation of results, transformation of the specific AHP procedure, number of hierarchy elements, provision of sensitivity analysis, learnability, user's effort needed for modifications, adaptation of problem structures, comprehensibility, availability of help, screen displays and initial cost. Therefore, to calculate priorities in the simulation strategy selection (SimSS) process, Expert Choice software was employed. In pair wise comparison, Expert Choice allows judgments to be entered either in numerical, graphical or verbal models. Verbal mode that consists of equal, moderate, strong, very strong, and extreme corresponding to 1, 3, 5, 7, and 9 in numerical scale.

4.3 The simulation strategy selection (SimSS) process

Based on the AHP solution process, the approach presented in figure 4.3 was derived to determine the most appropriate simulation strategy.

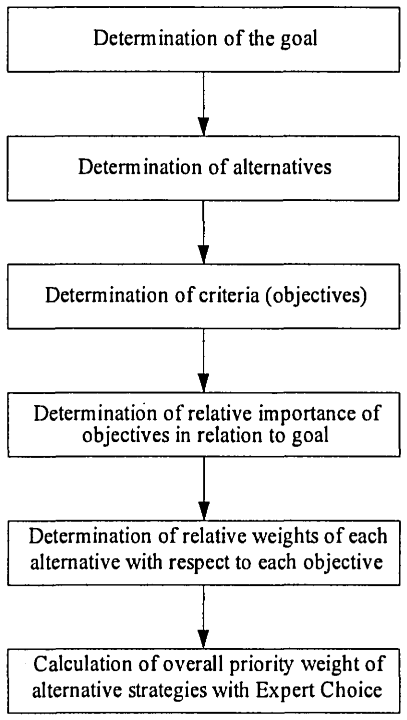

Determination of the goal
Determination of alternatives
Determination of criteria (objectives)
Determination of relative importance of objectives in relation to goal
Determination of relative weights of each alternative with respect to each objective
Calculation of overall priority weight of alternative strategies with Expert Choice

Figure 4.3 – SimSS process

The goal of the SimSS process is the determination of simulation strategy. Distributed simulation, parallel simulation and sequential simulation were selected as alternative strategies (figure 4.4). In some instances parallel simulation may provides an alternative to distributed simulation and vice versa. However, when simulation needs to be executed in a geographically distributed environment, the only viable option is distributed simulation.

4.3.1 Criteria for the SimSS process

Factors that encourage the application of parallel and distributed simulation technologies were used as the criteria (objectives) for the SimSS process. In order to identify these factors a sample of literature was analysed and the followings were identified as the widely cited ones.

- Execution time
- Parallelism
- Computational resources
- Geographic distribution
- Complicated model development
- Development time
- Fault tolerance

Fault tolerance was not considered for the SimSS process, as it is more associated with independent parallel replications of simulation than parallel and distributed simulation. Development time and complicated model development were grouped as complicated model development process. Therefore, the followings were selected as factors that encourage users to employ parallel or distributed simulation instead of sequential simulation.

Execution time

This indicates the time taken to run a simulation model. One of the main objectives of parallel or distributed simulation is to decrease the run time of a simulation. This form of simulation is expected to reduce the time taken to run a simulation with the aid of more computational resources and exploitation of the inherent parallelism.

Parallelism

In most of the simulation models, some sub-processes can be executed concurrently. Therefore, it is said to be that simulation models are inherently parallel. In parallel or distributed simulation, these parallely executable sub-processes are identified and partitioned into separate logical processes, and executed simultaneously. More computational resources and exploitation of inherent parallelism contribute to the speedup of simulations.

Computational resources

Computer memory, speed of the processor etc. may not be adequate to achieve target performances of a simulation. In addition, a single processor may not be capable to handle the entire simulation model of a complex system on its own. An obvious means of obtaining better performance is to dedicate more computational resources for simulation.

Geographical distribution

In some situations, sub-models of a simulation have to be run in geographically distributed locations. This may be due to availability of data, location of organization, management decisions etc. In this case, the only feasible alternative will be distributed simulation.

Complicated model development process

For many systems, especially large and complex systems, the model that characterizes the desired aspects of the system may itself be large and complex. Complicated models include more elements, interaction, detail etc. To construct a complicated model, services of more than one modeller may be required. Divide and conquer approach may provide a better alternative approach for complicated models. Partitioned sub-models are easier to comprehend, verify and validate, and convert into a computer program.

Execution time and computational resources were the most widely cited reasons. Execution time, lack of available resources and complicated model development process can be considered as constraints for sequential simulation while availability of more computational resources and ability to exploit parallelism act as motivators for moving into parallel or distributed simulation. Long simulation times, however are typically caused (at least partly) by lack of computational resources. The need to run a simulation in a geographically distributed manner is a deciding factor.

In order to simplify the process, exploitation of inherent parallelism was eliminated by assuming that parallelism is inherent to most of the real word problems. Based on this assumption and points noted previously in this section, following factors were selected as objectives for the SimSS process.

- Execution time
- Computational resources
- Complicated model development process
- Geographic distribution

Figure 4.4 shows the goal, objectives and alternatives for the SimSS process based on the AHP solution process.

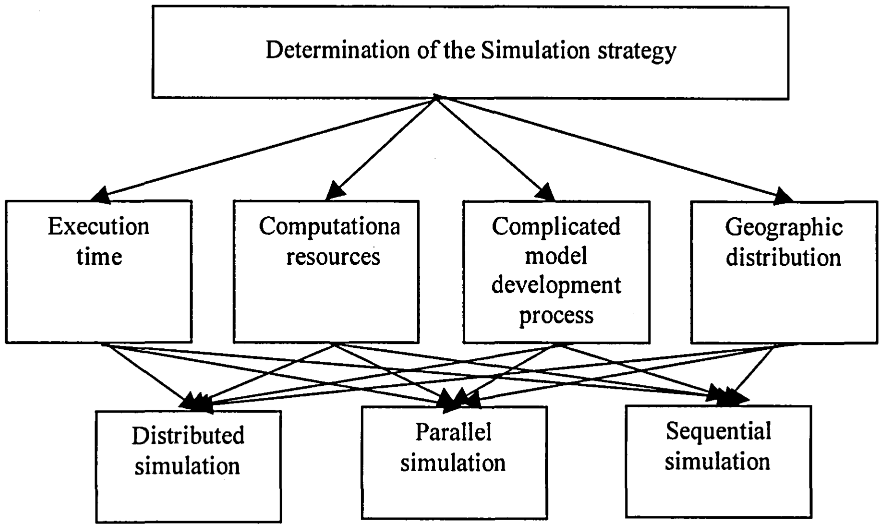


Figure 4.4 - Goal, objectives and alternative strategies

Three scenarios are presented in order to illustrate the SimSS process and to highlight the point that the decision on simulation strategy to be employed depends on the situation.

4.3.2 Illustration of the SimSS process

Scenario 1

Partners of an enterprise prefer to keep their part of the simulation model in their own premises, if possible. The model is complicated and difficult to develop as a single model. Execution time and computational resources are not critical factors. Tables 4.3 to 4.7 provide necessary pair wise comparisons for scenario 1.

Direction of the preference is indicated by direction of arrows.

Objective 1	Prefers to	Objective 2	Preference
Complicated model development	→	Geographic distribution	*
Computational resources	→	Geographic distribution	Extreme
Execution time	→	Geographic distribution	Extreme
Computational resources	→	Complicated model development	Very strong
Execution time	→	Complicated model development	Very strong
Execution time	↔	Computational resources	Equal

* Between equal and moderate

Table 4.2 - Pair wise comparison of objectives with respect to the goal

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	→	Parallel simulation	Moderate
Sequential simulation	→	Distributed simulation	Moderate
Parallel simulation	↔	Distributed simulation	Equal

Table 4.3 - Pair wise comparison to determine relative preference with respect to execution time

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	→	Parallel simulation	Moderate
Sequential simulation	→	Distributed simulation	Moderate
Parallel simulation	↔	Distributed simulation	Equal

Table 4.4 - Pair wise comparison to determine relative preference with respect to computational resources

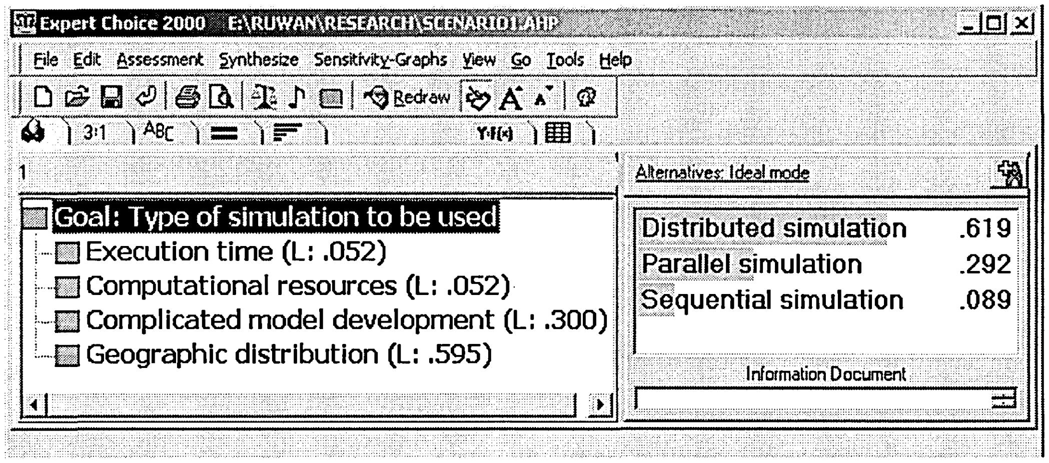
Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	→	Parallel simulation	Very strong
Sequential simulation	→	Distributed simulation	Very strong
Parallel simulation	↔	Distributed simulation	Equal

Table 4.5 - Pair wise comparison to determine relative preference with respect to complicated model development

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	$\leftrightarrow$	Parallel simulation	Equal
Sequential simulation	$\rightarrow$	Distributed simulation	Extreme
Parallel simulation	$\rightarrow$	Distributed simulation	Extreme

Table 4.6 - Pair wise comparison to determine relative preference with respect to geographic distribution

Figure 4.5 shows relative weights assigned to different objectives and priorities calculated in relation to the goal.


Expert Choice 2000 E:\RUWAN\RESEARCH\SCENARIO1.AHP
File Edit Assessment Synthesize Sensitivity-Graphs View Go Tools Help
Redraw
Alternatives: Ideal mode
Information Document

Goal: Type of simulation to be used
- Execution time (L: .052)
- Computational resources (L: .052)
- Complicated model development (L: .300)
- Geographic distribution (L: .595)

Distributed simulation	.619
Parallel simulation	.292
Sequential simulation	.089

Figure 4.5 – Priorities assigned to different alternative for scenario 1

Distributed simulation ranked as the strategy with highest priority. Therefore, the most appropriate simulation strategy for this scenario is distributed simulation.

Scenario 2

The model is complicated and difficult to develop as a single model. Execution time is longer than expected execution time and was suspected that lack of computational resources prolongs execution time. No specific need to run simulation in geographically distributed environment. Tables 4.8 to 4.12 provide necessary pair wise comparisons for scenario 2.

Direction of the preference is indicated by direction of arrows.

Objective 1	Prefers to	Objective 2	Preference
Complicated model development	←	Geographic distribution	Very strong
Computational resources	←	Geographic distribution	Very strong
Execution time	←	Geographic distribution	Very strong
Computational resources	→	Complicated model development	*
Execution time	→	Complicated model development	*
Execution time	→	Computational resources	*

* Between equal and moderate

Table 4.7 - Pair wise comparison of objectives with respect to the goal

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	→	Parallel simulation	Very strong
Sequential simulation	→	Distributed simulation	Very strong
Parallel simulation	↔	Distributed simulation	Equal

Table 4.8 - Pair wise comparison to determine relative preference with respect to execution time

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	→	Parallel simulation	Very strong
Sequential simulation	→	Distributed simulation	Very strong
Parallel simulation	↔	Distributed simulation	Equal

Table 4.9 - Pair wise comparison to determine relative preference with respect to computational resources

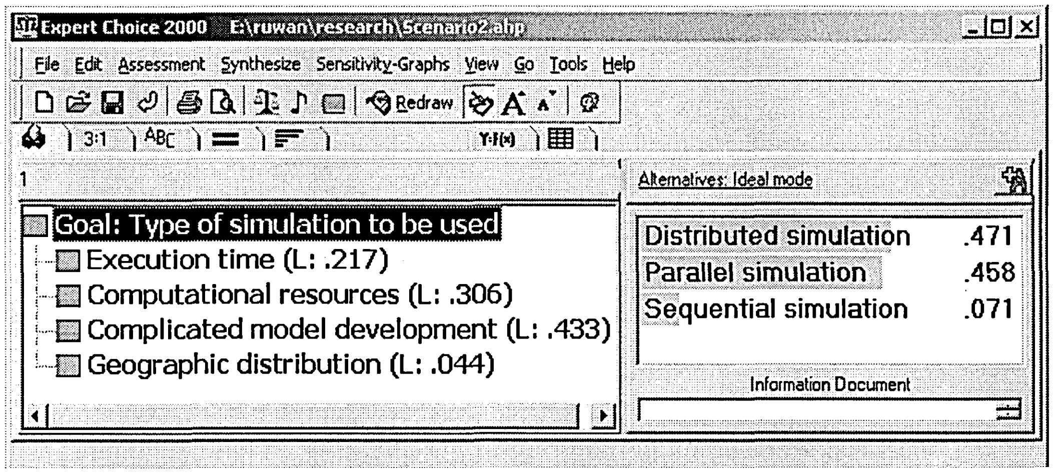
Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	→	Parallel simulation	Very strong
Sequential simulation	→	Distributed simulation	Very strong
Parallel simulation	↔	Distributed simulation	Equal

Table 4.10 - Pair wise comparison to determine relative preference with respect to complicated model development

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	↔	Parallel simulation	Equal
Sequential simulation	→	Distributed simulation	Extreme
Parallel simulation	→	Distributed simulation	Extreme

Table 4.11 - Pair wise comparison to determine relative preference with respect to geographic distribution

Figure 4.6 displays the relative weights assigned to different objectives and priorities calculated in relation to the goal.


Expert Choice 2000 E:\ruwan\research\Scenario2.ahp
File Edit Assessment Synthesize Sensitivity-Graphs View Go Tools Help
Redraw
3:1
Alternatives: Ideal mode
Information Document

Goal: Type of simulation to be used
- Execution time (L: .217)
- Computational resources (L: .306)
- Complicated model development (L: .433)
- Geographic distribution (L: .044)

Distributed simulation	.471
Parallel simulation	.458
Sequential simulation	.071

Figure 4.6 – Priorities assigned to different alternative for scenario 2

For this scenario distributed simulation or parallel simulation provides better alternatives. Distributed simulation has a slightly higher priority due to the fact that an extreme preference was assign to distributed simulation from parallel simulation with respect to geographical distribution

Scenario 3

The model is relatively a simple one when compared to models mentioned in scenarios 1 and 2. Available computational resources are satisfactory and there is no real need to speedup the simulation. No specific need to run the simulation in geographically distributed manner. Tables 4.13 to 4.17 provide necessary pair wise comparisons for scenario 3.

Direction of the preference is indicated by direction of arrows.

Objective 1	Prefers to	Objective 2	Preference
Complicated model development	←	Geographic distribution	Moderate
Computational resources	←	Geographic distribution	Moderate
Execution time	←	Geographic distribution	Moderate
Computational resources	↔	Complicated model development	Equal
Execution time	↔	Complicated model development	Equal
Execution time	↔	Computational resources	Equal

Table 4.12 - Pair wise comparison of objectives with respect to the goal

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	←	Parallel simulation	Moderate
Sequential simulation	←	Distributed simulation	Moderate
Parallel simulation	↔	Distributed simulation	Equal

Table 4.13 - Pair wise comparison to determine relative preference with respect to execution time

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	←	Parallel simulation	Moderate
Sequential simulation	←	Distributed simulation	Moderate
Parallel simulation	↔	Distributed simulation	Equal

Table 4.14 - Pair wise comparison to determine relative preference with respect to computational resources

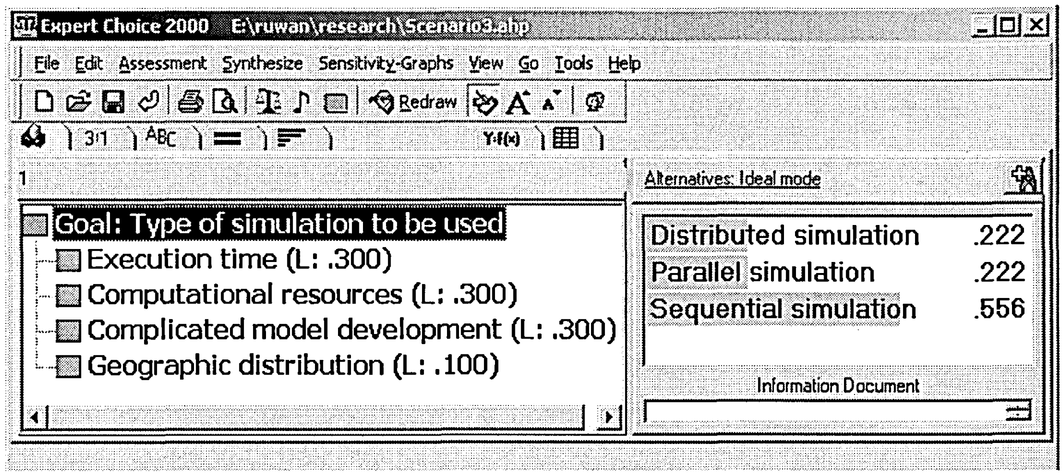
Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	←	Parallel simulation	Moderate
Sequential simulation	←	Distributed simulation	Moderate
Parallel simulation	↔	Distributed simulation	Equal

Table 4.15 - Pair wise comparison to determine relative preference with respect to complicated model development

Alternative 1	Prefers to	Alternative 2	Preference
Sequential simulation	↔	Parallel simulation	Equal
Sequential simulation	↔	Distributed simulation	Equal
Parallel simulation	↔	Distributed simulation	Equal

Table 4.16 - Pair wise comparison to determine relative preference with respect to geographic distribution

The relative weights assigned to different objectives and priorities calculated in relation to the goal are shown in figure 4.7.


Expert Choice 2000 E:\ruwan\research\Scenario3.ahp
File Edit Assessment Synthesize Sensitivity-Graphs View Go Tools Help
Redraw
Alternatives: Ideal mode
Information Document

Goal: Type of simulation to be used
- Execution time (L: .300)
- Computational resources (L: .300)
- Complicated model development (L: .300)
- Geographic distribution (L: .100)

Distributed simulation	.222
Parallel simulation	.222
Sequential simulation	.556

Figure 4.7 – Priorities assigned to different alternative for scenario 3

In this scenario, the appropriate strategy would be sequential simulation.

Summary

This chapter presented a new approach to select an appropriate simulation strategy from parallel simulation, distributed simulation or sequential simulation, as parallel and distributed simulation is not suitable for all simulation problems. It was illustrated that there is no one best simulation strategy for all situations and the appropriate simulation strategy depends on the situation. If it is determined that distributed (or parallel) simulation to be employed, then one has to move into the next step of the proposed methodology for distributed enterprise simulation, namely the data collection stage. However, this stage is well researched area in the simulation literature and was briefly described in chapter 3. The following chapter presents the next two stages of the proposed methodology which involve developing the conceptual model, partitioning the conceptual model and assigning partitioned logical processes to networked workstations.

Chapter 5
Conceptual modelling, model partitioning and mapping for distributed enterprise simulation

The last chapter presented the simulation strategy selection (SimSS) process which helps to determine the appropriate simulation strategy from sequential simulation, parallel simulation or distributed simulation. If distributed simulation is chosen as the appropriate simulation strategy, then the system under investigation needs to be partitioned into sub-models or logical processes, and assigned to geographically distributed workstations. This chapter presents a systematic approach for conceptual modelling, model partitioning and mapping for distributed enterprise simulations (see the highlighted activities in figure 5.1). It pays more attention to conceptual modelling than model partitioning and mapping as it is proposed to partition the conceptual model before transforming the model into a computer simulation model, which is another main difference between some of the existing approaches and the proposed methodology.

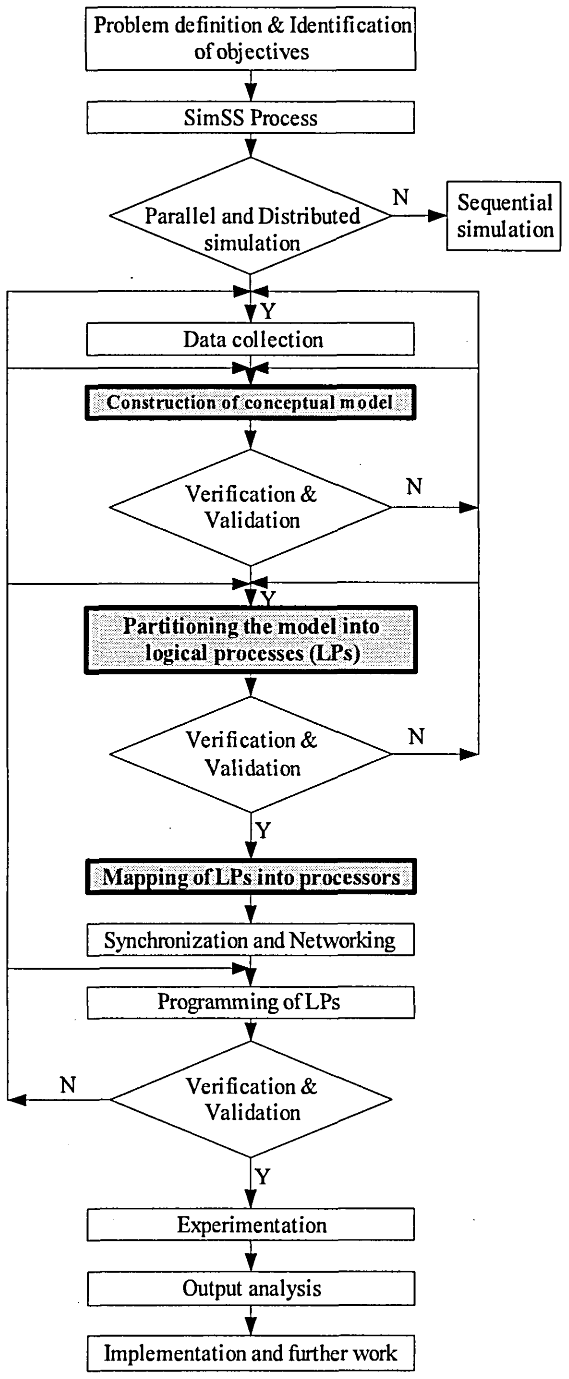

Problem definition & Identification of objectives
SimSS Process
Parallel and Distributed simulation
N
Sequential simulation
Y
Data collection
Construction of conceptual model
Verification & Validation
N
Y
Partitioning the model into logical processes (LPs)
Verification & Validation
N
Y
Mapping of LPs into processors
Synchronization and Networking
Programming of LPs
N
Verification & Validation
Y
Experimentation
Output analysis
Implementation and further work

Figure 5.1 - The proposed methodology for distributed enterprise simulation

5.1 Introduction

Modelling is an essential part of any simulation project, including distributed simulation which provides the foundation for distributed enterprise simulation. As noted in chapter 4, the main difference between sequential simulation and distributed simulation is: while sequential simulation runs as a single model in a single workstation, in distributed simulation several models run in geographically distributed (and interconnected) workstations. Accordingly, for a simulation to run in a geographically distributed environment, the entire simulation model has to be partitioned into a number of logical processes (LPs) or sub-models and assigned (mapped) them to different workstations.

One of the most important issues to be addressed when designing a distributed simulation including enterprise simulation is the partitioning of the simulation model into several LPs. Efficiency and effectiveness of a distributed simulation system depend on partitioning of the system. Performance of distributed simulation will be detrimentally affected if the workload of one LP is significantly higher than the others. Furthermore, increasing the load on the network may result in slowing down other applications that run across the network if frequency of interactions between (two) LPs assigned to different workstations is high. Once the partitioning process is completed, the resulting LPs need to be assigned to different processors, which is known as mapping. In distributed simulation, LPs are assigned to processors, which reside on geographically distributed workstations. One or more LPs can be assigned to a single processor in order to balance the workload among processors. When compared to issues such as synchronization, the literature on (parallel and) distributed simulation has not paid much attention to conceptual modelling, model partitioning and mapping. Moreover, partitioning and mapping algorithms presented in the literature are generally complex and some of the algorithms require running of the simulation code sequentially in order to identify LPs. A simulation is executed in a distributed manner because of its inability to run sequentially due to the size, complexity, requirements for more computing resources, or specifically needs to run in geographically distributed environment. This creates a dilemma for users especially in business organizations, who intend to employ distributed simulations. Therefore it is desirable to have a simple yet effective approach for model partitioning and mapping when developing distributed enterprise simulation.

The objective of this chapter is to present a new approach for conceptual modelling, model partitioning and mapping for distributed enterprise simulation. The new approach proposes to partition the conceptual model developed for the system under investigation and then map them onto the processors of geographically distributed workstations. The next section provides a brief description of the conceptual model. Modelling approaches and modelling tools are presented in section 5.3 and 5.4 respectively. Model partitioning and mapping approaches are briefly explained in the following section. Section 5.6 presents the proposed approach for model representation, model partitioning and mapping. The chapter ends with a summary.

5.2 The conceptual model

A model is an abstract representation of reality (Whitman et al., 1997). The degree to which the simulation results are able to characterize the system under study is directly related to the degree the simulation model characterizes the system (Luna, 1992). For many systems especially complex and large ones, it is desirable to build a conceptual model before transforming it into a computer simulation model in order to understand the problems, requirements and perhaps alternative solutions. Borah (2002) defined the conceptual model as an abstract representation of something generalized from particular instances. A conceptual model is a simulation developer's way of translating modelling requirements (i.e. what to be represented by simulation?) into a detailed design framework (i.e. How it is to be done?), from which the software that will make up the simulation can be built (Pace, 1999). It can be utilized as a means of clear and comprehensive communication among developers of simulation, managers, users and other stakeholders. Furthermore, the conceptual model is the ultimate expression of the system functionality and should be the basis for testing, verification and validation procedures (Haddix, 2001). Firat (2000) summarized the functions of conceptual models as:

- Providing a means for verification and validation
- Improving understanding when analyzing new and old systems
- Providing a precise and clear tool for communication between system developers and application domain specialists
- Providing documentation of domain specific information in a formal way
- Making easier later modifications on the system after development since it is the blueprint of the system
- Allowing determination of conflicts among different perspectives of requirements for the same system and helps to elicit conflicts

Building of simulation models has long been considered as an art rather than a science (Leung and Lai, 1997). Karacal (1998) also noted that despite the existence of well-developed tools and their generalized building blocks, modelling is still carried out in an ad hoc and intuitive manner. Although construction of a conceptual model is important when developing sequential simulations, special attention needs to be paid when building distributed simulations as this type of simulations are more complex than the former. Past research clearly demonstrated a need for developing innovative modelling methods and procedures that will assist in the development of simulation models for large and complex systems (Sawhney, 2000). Furthermore, many authors highlighted the need for formal simulation model building methodologies for distributed (also parallel) simulation (Brandimarte and Cantamessa, 1995; Karacal, 1998; Odhabi et al., 1997; Page, 1999).

Proper development of a conceptual model is critical as it describes how a simulation developer intends an implementation to satisfy requirements. This model is the primary mechanism for transforming simulation requirements into specifications that can guide simulation development and implementation process (figure 5.2). Therefore special attention is needed when developing the conceptual model. A series of articles presented in recent simulation interoperability workshops highlighted the importance of conceptual modelling and provided guidance on development and evaluation of conceptual models (Borah, 2000; Borah, 2002; Firat, 2000; Haddix, 2001; Pace, 1999 and 2000).

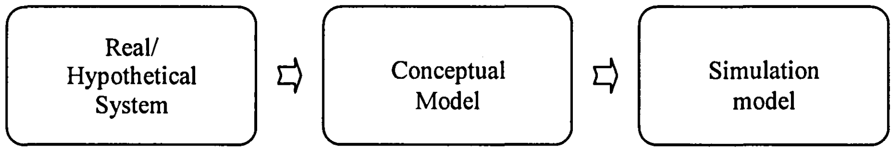


Figure 5.2 – Significance of the conceptual model

When developing a conceptual model, a modelling approach and modelling tools have to be determined beforehand. Modelling approaches specify the way models are to be developed. Once the approach is determined, the modeller can decide what modelling tools are to be used for model building. The resulting model then can be transformed into a simulation model. This confirms that simulation involves more than merely writing of a computer program, as highlighted by Page and Nance (1994). To overcome the challenge of modelling complex systems, a number of modelling approaches have been proposed. Most of these approaches can be classified under incremental modelling and hierarchical modelling approaches.

5.3 Modelling approaches

5.3.1 Incremental modelling approach

This approach is based on incremental development of a model with few elements and little detail, capturing a holistic view of the system under investigation. The model is therefore initially not large, but might become so as development progresses. Pidd (1996) suggested that starting with a small model and adding more details is one way to ease the difficulties of building models for complex systems. Randell et al. (1999) mentioned that modularization is a prerequisite for incremental model development. Modularization reduces the complexity, and allows modelling at a higher level of abstraction. Pidd and Castro (1998) also noted that modularity is the key to coping with the complexity inherent in large systems.

Under the incremental model development approach, two ways of dividing the simulation project into stages can be identified (Randell et al., 1999). One is to work vertically first and then horizontally. The other way is to take a holistic view of the system and develop a model, and then add detail later as required. The latter approach shares common characteristics with the hierarchical modelling approach, which will be discussed later. The top down approach minimizes the risk of sub optimization. Also only subsystems that need to be analyzed further have to be added to the main model, leaving others as black boxes. In addition, the incremental modelling approach spreads model development cost over the life-time of the project.

5.3.2 Hierarchical modelling approach

Design of complex systems is confronted with a problem of describing system objects, their characteristics and interactions in a concise and understandable way (Ceric, 1994). One of the basic strategies for accomplishing this task is the hierarchical approach. Furthermore, Pidd and Castro (1998) noted that many large systems are inherently hierarchical. A hierarchy essentially defines a type of relation in which the entities are grouped at different levels. Chow and Zeigler (1994) pointed out that hierarchical modelling capability is increasingly being recognized as the predominant modelling paradigm for future simulation developments. Hierarchical modelling develops model elements from higher levels into a more detailed description on lower hierarchical levels. It provides a way of managing large scale complex systems by considering them as a collection of sub-systems which are represented by simulation models that are independently created, modified and saved (Kiran, 1998).

The model to be simulated depends on decisions relating to the level of abstraction of the system. The correct level of abstraction refers to selecting the amount of information that must be included in the model to help address the modelling goals (Benjamin et al., 1998). Decomposition (dis-aggregation) and abstraction (aggregation) are two important principles of hierarchical modelling. Decomposition refers to adding more details to a selected level of abstraction resulting of a model with lower level of abstraction. Aggregation refers to summarizing information of a selected level of abstraction, resulting in of a model with a higher level of abstraction (figure 5.3). It reduces the number of components and interactions of the model thus reducing the overall behavioural complexity (Fishwick, 1994). Ball (1998) noted that use of an appropriate level of details (level of abstraction) allows building simplified models, which run faster.

In addition, Ceric (1994), Sargent et al. (1993) and Zupancic (1998), highlighted a number of advantages of hierarchical decomposition in the hierarchical modelling approach, including:

- Possibility of focusing on each component as a small problem
- Several modellers can work simultaneously on a modelling a simulation project
- Information hiding
- Improved communication with users
- Easy to implement modifications and corrections
- The modular structure enables partial testing of the model
- Easy to document the system
- Enables the application of different algorithms to different sub-systems
- Reduce effort and time required to develop models
- Allow developing sub-models separately and integrating later permitting model reusability
- Assist in model verification and validation process.

Furthermore, hierarchical modelling helps development of distributed simulations by identifying sub-systems (LPs) that can be functioned independently. These LPs can be later assigned to different workstations to run the simulation system in a geographically distributed environment.

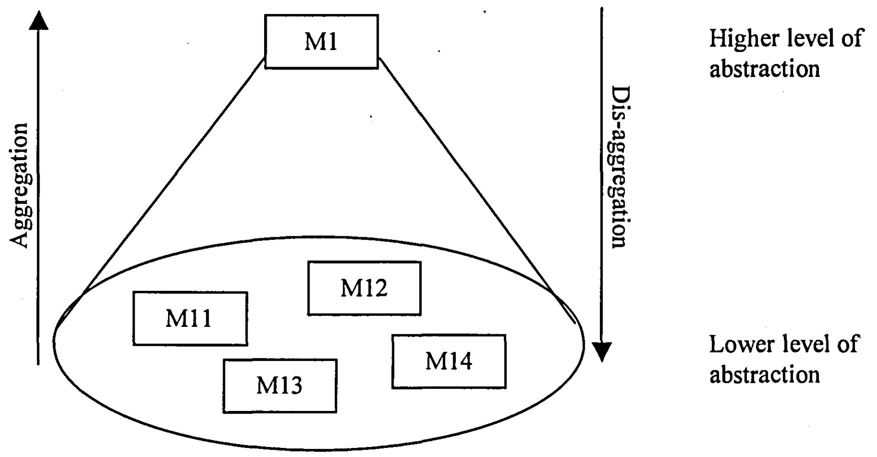


Figure 5.3 – Level of abstraction

Hierarchical modular modelling (Luna, 1992; Pidd and Castro, 1998; Sawhney, 2000; Zeigler, 1986; Zupancic, 1998) is a variation of hierarchical modelling. In addition to the hierarchical approach to model building, it proposes model building by adding components/ modules. This allows different components to be developed, verified and validated separately (Luna, 1992). It also encourages building of modules simultaneously resulting in of shorter development times and cost savings. Alfieri and Brandimarte (1997) and Zeigler (1987) proposed the use of an object-oriented approach in hierarchical modular modelling. An advantage of using this approach is that there is a better transition from modelling concepts to actual software implementation, since objects have a natural match in the real world (Alfieri and Brandimarte, 1997).

5.4 Modelling tools

Modelling tools provide a standard means of describing and analyzing a system. This facilitates communication between developer and user, and between developers. It also simplifies understanding, modification and maintenance of systems, ensuring good discipline. Pandya (1995) described a modelling tool as a communication device that is used to aid generation and classification of ideas, and/or to analyze the quality of a design. A number of modelling tools are available to develop a model of a new or existing system. Some of the well known tools are described below.

5.4.1 Diagrammatic modelling

Diagrammatic models are a particular class of conceptual models which enable graphical representation of models in two dimensions. This approach uses symbols to represent physical elements and activities of the system under investigation, and directed arrows to indicate the direction of flow. In the analysis phase of a simulation study, the graphical representation approaches serve as a very useful framework with which the modeller can analyze and conceptualize the problem and as a communication medium among the people who are involved in the project (Kienbaum and Paul, 1994). Ceric (1994) noted that diagrammatic modelling methods are one of the most used and developed class of conceptual modelling methods in discrete event simulation. Reasons that bought popularity to diagrammatic modelling include:

- Conceptually close objects can be represented as physically adjacent, bringing to light the strength of connection in the system.
- Interactions between objects are shown in two dimensions, enabling much easier comprehension of a model than the forced sequential ordering of objects in procedural representations. This is due to a parallelism of human visual system which enables fast visual processing of the whole model or its significant parts.
- Syntax and semantics of diagrammatic modelling methods are often rather simple, which helps the easier and faster model design and understanding.
- Hierarchical model decomposition is possible in most diagrammatic methods, which again assists both modelling of complex systems and model understanding.
- Most diagrammatic models enable manual simulation of system dynamics. This feature can help in model validation, and also useful as a simulation learning tool too.

Furthermore, diagrammatic structure of the model looks similar to structure of the simulation model especially if the simulation model is developed using commercial simulation software.

In respect to distributed simulation, it is vital to use formal modelling tools to develop the conceptual model due to its complex nature. In addition, they are even more attractive for the proposed approach for distributed enterprise simulation as it is proposed to use commercial simulation software for implementation. Some of the commonly used diagrammatic modelling tools in simulation are briefly described below.

5.4.2 Commonly used diagrammatic modelling tools

Activity cycle diagram

Activity Cycle Diagrams (ACD) have long been used for representation of the flow of entities within discrete event systems. Apart from using as a model representation tool, ACDs can also be used to manually simulate the system. Original ACDs make use of only two symbols: a circle to represent a dead state and a rectangle to represent a live state (figure 5.4). The diagram itself is a map which shows the life history of each class of entity and graphically displays their interactions. Each class of entity is considered to have a lifecycle which consists of a series of states. The entities move from state to state as their life proceeds.

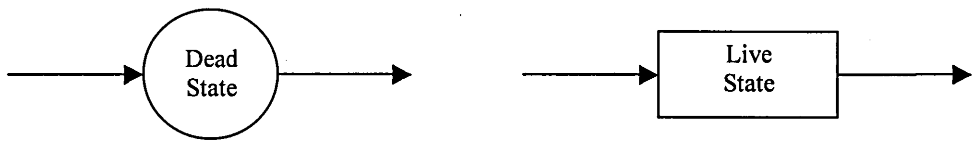


Figure 5.4 - Symbols of original ACD

The main advantages of ACDs include simplicity, ease of understanding and support of hierarchical modelling approach. However, Pflughoeft and Manur (1994) mentioned that advantages of ACD in its original form were outweighed by inefficiencies. ACD of a complex system is too complicated and cumbersome for its intended purpose. The simplicity of ACDs and their associated limitations for developing computer based simulations for complex systems motivated a number of authors to present modified versions of ACDs. Pooley (1991) proposed an extended set of symbols to represent processes of a simulation. This modified version of ACD was called as Extended Activity Cycle Diagrams (X-ACDs). Kienbaum and Paul (1994) presented Hierarchical Activity Cycle Diagrams (H-ACDs) that support object oriented simulation modelling. Pflughoeft and Manur (1994) introduced Multi layered ACD approach which decomposes the diagram by activities, instead of entity flows.

Although analysis of papers presented to recent Winter Simulation Conferences (WSC) shows that number of papers published declined over the past few years, a number of authors including Baldwin et.al. (2000), Eldabi and Paul (2001), Odhabi et al. (1997), Odhabi et al. (1998) and Shi (1997) presented papers on applications of ACD.

Petri nets

Petri nets are graphical and mathematical modelling tools that can be used to perform static and dynamic analysis of processes that constitute existing or new systems. The concept of Petri nets originated from works of Carl A. Petri in 1962.

Petri nets are used for describing and studying systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic and stochastic (Sawhney et al., 1999). Graphical modelling elements of Petri nets are shown in figure 5.5.

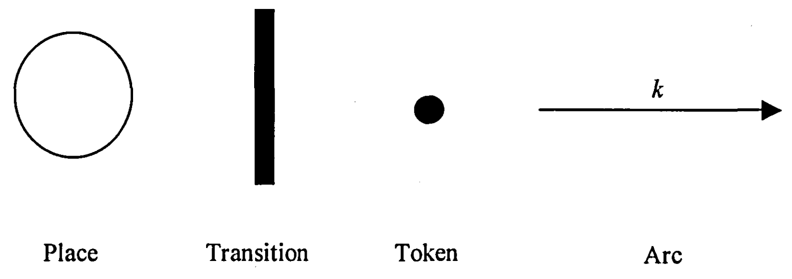


Figure 5.5 - Graphical elements of Petri nets

A place denoted by a circle represents a condition such as input data, input signal, resource, condition, or buffer. A transition denoted by a solid bar represents an event such as a computational step, task or activity. Arcs are used to connect places and transitions in a Petri net. They are directed and are either drawn from a place to a transition or from transition to a place. Arcs in a Petri net can also have multiplicity which is represented by an integer *k*. Multiplicity indicates the number of tokens required to fire or enable a transition. Token which is denoted by solid small circle provides the dynamic simulation capabilities to Petri nets. Without a provision of tokens in a Petri net, the dynamic behaviour of the system under consideration can not be simulated and the Petri nets can only be used as a visual communication tool.

As with ACDs, several variations to the classical Petri nets such as Timed Petri Nets (TPN), Coloured Petri Nets (CPN), High-level Petri Nets (HPN) were presented by a number of authors (Choila and Ferscha, 1993; D'Souza and Khator, 1994; Gerogiannis et al., 1998; Gile and DiCesare, 2001; Vojnar, 1997). Pandya (1995) mentioned that Petri nets strike a balance between the speed and simplicity of mathematical programming and the flexibility provided by general purpose simulation packages. However, he also highlighted the following problems associated with Petri nets too.

- Lack of general methodology for constructing Petri nets models from system specifications
- Diagrams can become cluttered when modelling complicated systems
- Lack of general software to support the computer coding of Petri net models

IDEF0

IDEF is a system definition method developed under sponsorship of the US air force to describe information and structure of complex manufacturing systems. The acronym IDEF stands for ICAM DEFinition where ICAM stands for Integrated Computer Aided Manufacturing. IDEF is not a single technique and is a family of techniques extending from IDEF0 to IDEF5 including IDEF1x. IDEF0 was derived from a well-established graphical language, the Structured Analysis and Design Technique (SADT) and used as a functional modelling tool for analyzing and communicating the functional perspective of a system. Main elements of IDEF0 composed of a box and an arrow. Boxes are used to represent system functions and, data or object interfaces are represented by arrows (see figure 5.6).

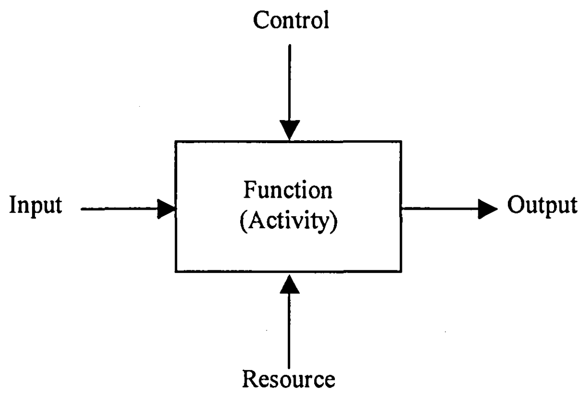


Figure 5.6 – Elements of IDEF0 technique

- An arrow coming into a box from left depicts input required to perform the function
- An arrow coming out of a box on the right depicts output produced by the function
- An arrow coming into a box from top shows controls that represent conditions, circumstances or rules by which the function is driven

- An arrow coming out of a the box from bottom show physical resources or mechanism required to perform the function

The generation of many levels of details through the model diagram is one of the most important features of IDEF0 as a modelling technique. The IDEF0 model starts by representing the whole system as a single box (which is labelled as A0). The A0 box then can be broken down into more detailed diagrams until the system described in the desired level of detail. As IDEF0 modelling technique supports hierarchical modelling approach, an abstracted system can be decomposed into a more detailed set of diagrams in a hierarchical manner as shown in figure 5.7.

Pandya et al. (1997) summarized the following benefits and shortcomings of IDEF0.

Benefits

- Modelling of large and complex systems made possible by decomposing an abstracted level of the system into more detailed level as desired.
- Easy to understand as only few symbols (boxes and arrows) are used to model the system.
- Distinguishes between input, output, controls and resources for a particular activity.
- Notation of the model allows an easy development of computer support.
- Existence of well documented rules and procedures

Shortcomings

- Only provides a static representation of the system
- Does not take time and cost to perform an activity into account
- Does not make a distinction between data and material flow

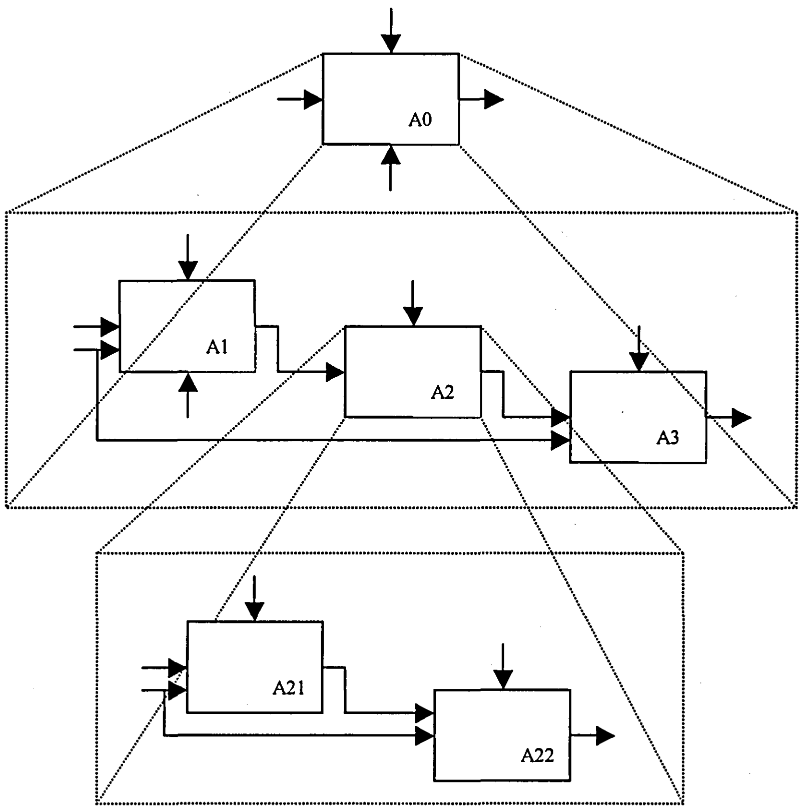


Figure 5.7 – Hierarchical decomposition in IDEF0

Other modelling tools

In addition to modelling techniques described earlier, a number of techniques such as activity diagrams, GPSS block diagrams, event graphs etc. can be used to develop a conceptual model. More details of these techniques are presented by Buss (1996), Ceric (1994), Pooley (1991) and Schruben (1983). Analysis of articles presented to the Winter Simulation conferences suggest that number of articles publishedana on these techniques declined over past few years.

5.4.3 Modelling methods

Modelling methods propose methodological approaches for modelling. These methods are generally used to design new systems, study existing systems etc. In addition to some of the modelling techniques mentioned earlier (such as ACD and IDEF0), data flow diagrams, entity relation diagrams etc. are used as tools to model systems. For developing a conceptual model for a simulation, these methodologies are not required to use fully. Howver, it is desirable to use some of the procedures prescribed in them in order to improve the accuracy to the conceptual model. Widely cited modelling methods in the literature include SSADM (Structured System Analysis and Design methodology), SADT (Structured Analysis and Design technique) and GRAI (Graph with Results and Actions Interrelated) methodology (Al-Ahmari and Ridgeway, 1999; Doumeingts et al., 1995; Kateel et al., 1996; Pandya, 1995; Pandya et al., 1997).

5.5 Model partitioning and mapping approaches

Improved performance gained by distributing a simulation system into multiple processors is largely determined by how well the entire system is divided between processors. The problem of partitioning and mapping involves grouping and assignment of LPs to processors in such a manner that the communication overhead is minimized and the processor utilization is maximized (Nandy and Loucks, 1992).

There are two approaches for dividing the entire system into a set of sub models (Luksch, 2002 and Nutt, 1990). In functional partitioning, the simulation model is partitioned based on functions performed by the simulation system such as random number generation, input data and output data handling etc. In a model or data partitioning approach the system being simulated is partitioned into a number of sub-models which will be able to be executed in parallel. According to Fujimoto (1990), model partitioning approach is generally used for parallel and distributed simulation and the partitioned sub-models are known as LPs. As the size of the of the LPs decreases, the ability of distributed simulation models to run concurrently improves, resulting in higher levels of speedup. However, the desired level of performance improvements may not be achieved due to the increased load on the network as a result of more messages need to pass for synchronization (Hao et al., 1996) and also for passing parameters between distributed simulation models. Therefore both communication overheads and total execution time have to be carefully considered when a simulation model is partitioned and mapped. Another important factor to be considered is balancing the load of the distributed simulation by uniformly distributing the execution load among the processors.

In the literature the term partitioning has been used for decomposition and allocation of LPs to a network of processors. Luksch (1995) and Nutt (1990) used the term "data partitioning" for model decomposition. However, Boukerche and Tropper (1994 and 2001) and Szynkiewicz (2000) considered partitioning as allocating LPs to processors. For the purpose of decomposing and allocating LPs to processors, the following approaches presented in Boukerche and Tropper (2001) can be employed.

Random partitioning

With this simple approach, LPs are assigned to processors randomly. Although this algorithm is easy and fast to implement, the outcome may be poor. This is due to non-consideration of inter-processor communication. High communication overhead may slow down the simulation.

Grid partitioning

The grid partitioning approach reduces the communication overhead by combining communicating LPs together. In this algorithm, the process graph is sub-divided into grids, and all of LPs in the same grid are allocated to the same cluster.

Strongly connected component partitioning

This approach improves the grid partitioning algorithm by considering the inter-processor communication overheads and the possibility of inter-processor deadlocks.

In addition, a number of authors including Boukerche and Fabbri (2000), Boukerche and Tropper (1994), Choila and Ferscha (1993), Cloutier et al. (1997), Hendrickson and Kolda (2000) and Kim et al. (1998) described and discussed different partitioning and mapping algorithms.

5.6 Proposed approach for model representation, model partitioning and mapping

As was already noted, most of the distributed simulations were developed with general purpose or special simulation languages by employing partitioning and mapping algorithms to decompose and assign LPs to workstations (processors). However, partitioning algorithms proposed in the literature (Boukerche and

Fabbri, 2000; Boukerche and Tropper, 1994 and 2001; Cloutier et al., 1997; Hendrickson and Kolda, 2000; Kim et al., 1998; Nandy and Loucks, 1992) are complex and difficult to implement without a higher level of expertise in parallel and distributed simulation (especially in partitioning), computer programming and mathematics. Moreover, most of these authors did not comment on how to implement their proposed algorithms. Some partitioning algorithms including one proposed by Nandy and Loucks (1992) require the distributed simulation program to run initially as a sequential simulation. This is done to calculate the execution times of different elements, frequency of communication between them and the time taken to pass messages. Once the required information is collected, the simulation model is partitioned into LPs and mapped onto different processors. However, generally in the literature it is not clear how LPs are identified when the simulation model is developed. Thus, the simulation community, especially from business organisations may find it difficult to implement distributed simulations. Although most of the literature in distributed simulation does not specifically mention how to construct simulation models to execute in distributed environment, it can be presumed that figure 5.8 generally summarises the existing approaches.

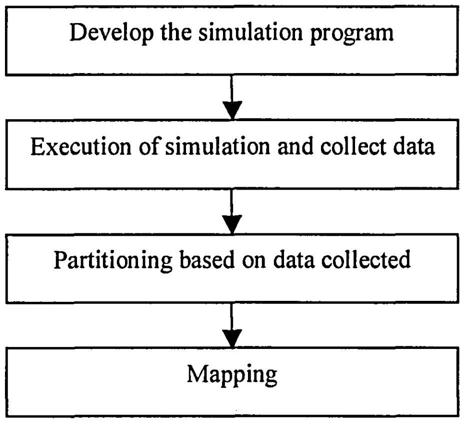

Develop the simulation program
Execution of simulation and collect data
Partitioning based on data collected
Mapping

Figure 5.8 – Existing approaches for conceptual modelling, model partitioning and mapping

Simulations are executed in a distributed manner mainly because the simulation model is too large or too complicated to be executed in a single processor (workstation). This is especially relevant for enterprise simulation where process sequences are often complex and number of resources employed are large

(Sirinivasan and Jayaraman, 1997). Furthermore, a simulation model may be developed at different sites and then linked together to be executed as a distributed simulation. Also different sections or partners of the enterprise may not want to share information with other sections or partners. Therefore, partitioning and mapping techniques mentioned previously are not appropriate for the proposed approach as it is necessary to partition the conceptual model before transforming it into a computer simulation model.

As noted earlier, the main difference between existing approaches and the proposed approach is the point of partitioning carried out in the simulation methodology. According to most of the current approaches partitioning is done after the system is converted into a computer program using algorithms in order to minimize the communication overheads and optimize the load balance. To simplify the distributed simulation development process it was proposed to partition the conceptual model into LPs, assign them into processors and then transform LPs into computer simulation models. Furthermore, as functions and sub-functions can be easily identified in an enterprise, the proposed approach can be easily applied when developing a distributed enterprise simulation than constructing distributed simulations for highly complex systems such as logic circuits, computer networks, telecommunication systems etc.

For the purpose of the proposed approach for model representation, partitioning and mapping of enterprise simulations, a LP can be described as follows:

- A single business entity of the enterprise
- A function of a business entity
- A sub-function

An enterprise could be partitioned in such a way that LPs could function independently of each other and continue to serve "local" needs of the business functions they represent (Datar, 2000). In this case a simulation model already developed for a section could be used to simulate that section of the enterprise and with appropriate modifications could also be connected to simulation models that represent other sections of the enterprise to simulate the whole enterprise. This could easily be carried out by keeping simulation models for different sections in different physical locations which they represent and running them in a distributed manner. The key to design of such a system is to identify sections of an enterprise that could function independently. The selected modelling approach and modelling tool may play a critical role in decomposing an enterprise into different sections that could function in parallel at a selected level of abstraction.

The hierarchical modelling approach was selected since it provides a way of managing large scale complex systems by considering them as a collection of sub-systems (Kiran, 1998). In a distributed simulation system these are represented by the simulation models that are independently created, modified and saved.

IDEF0 was selected as the modelling technique for the proposed approach for conceptual modelling, model partitioning and mapping. IDEF0 is simple and able to support different abstraction levels. It has been widely used due to its user-friendliness, computer support, rigor and conciseness, and well documented rules and procedures (Kateel et.al., 1996). Pandya (1995) noted that IDEF0 has been widely used in industry, resulting in the existence of a wide user base. A number of authors including Cheng-Leong et al. (1999), Cheng-Leong (1999), Rensburg and Zwemstra (1995) and Whiteman et al. (1997) have used it as a model representation technique in simulation. Another benefit of using IDEF0 with commercial simulation software is that the IDEF0 structure of the model can easily be transformed into a simulation model. Figure 5.9 shows a part of simulation model developed by Arena for an IDEF0 model. This helps to reduce the complexities associated with development of simulation models particularly distributed simulations which, according to the literature are more complicated to develop.

With the hierarchical modelling approach and the IDEF0 technique, LPs that can function independently could be identified based on interactions between different sections. In the IDEF0 model these interactions are represented by lines between boxes that represent different sections of the enterprise.

Once sub-models are identified, this could be validated to make sure that the sub-models represent the enterprise when taken as whole. Then the validated sub-models could be mapped out to processors in a network of workstations before being converted into computer simulation models and executed as a distributed enterprise simulation. In order to simplify the mapping processes and assuming that networked workstations are freely available to assign LPs, it is proposed that only one LP is mapped into a (processor of) workstation.

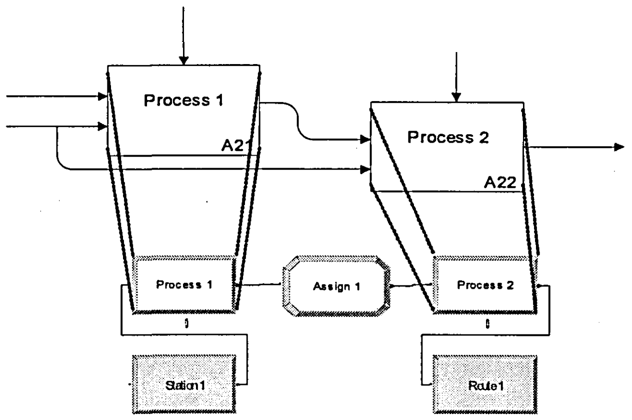


Figure 5.9 – Relationship between IDEF0 diagram and Arena simulation model

Based on the ideas presented above, the following approach was proposed for the purpose of conceptual modelling, model partitioning and mapping for distributed simulation in order to execute enterprise simulation models (figure 5.10).

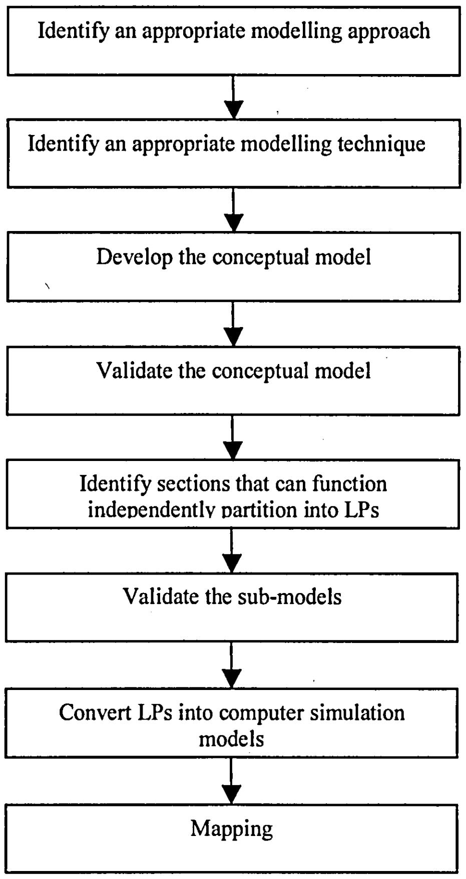

Identify an appropriate modelling approach
Identify an appropriate modelling technique
Develop the conceptual model
Validate the conceptual model
Identify sections that can function independently partition into LPs
Validate the sub-models
Convert LPs into computer simulation models
Mapping

Figure 5.10 – Proposed approach for model representation, model partitioning and mapping

The conceptual model needs to be validated in order to make sure that the conceptual model represents the enterprise as intended. After partitioning, sub-models should be again validated to ensure that when integrated they represent the system under investigation. The accuracy of simulation can be also improved with this step by step validation approach.

Summary

This chapter presented a simplified approach for model representation, model partitioning and mapping for distributed enterprise simulations. Before transforming sub-models into computer simulation models, a synchronization protocol needs be determined as the programming code for synchronization is integrated into simulation models. The next chapter addresses synchronization and networking issues in distributed simulation and presents a synchronization mechanism which focuses on distributed manufacturing applications.

Chapter 6

The proposed synchronization mechanism for the distributed enterprise simulation

Chapter 5 presented a discussion of conceptual modelling, model partitioning and mapping. Before transforming the partitioned logical processes into computer simulation models and executing them in a distributed simulation environment, the infrastructure required for distributed simulation and a synchronization approach have to be determined. This chapter addresses networking and synchronization issues relating to distributed enterprise simulation (figure 6.1). It presents brief descriptions on network topologies, communication protocols and network protocols, synchronization and different synchronization protocols. An approximate synchronization mechanism is proposed as an alternative for strictly synchronized approaches.

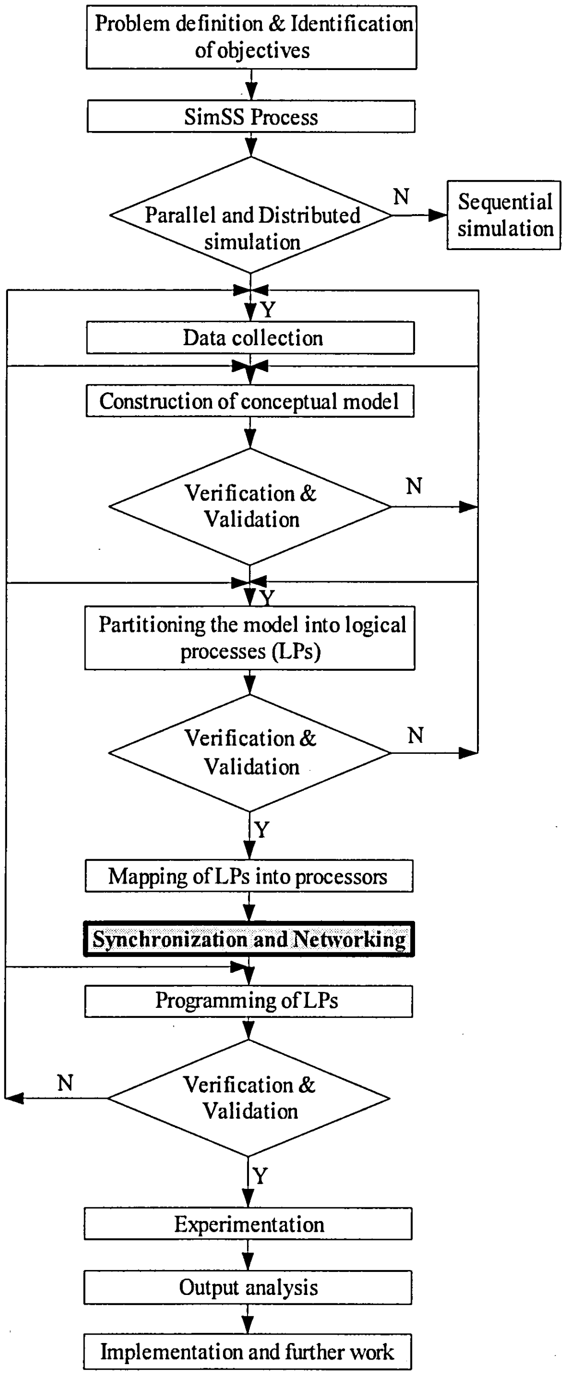

Problem definition & Identification of objectives
SimSS Process
Parallel and Distributed simulation
N
Sequential simulation
Y
Data collection
Construction of conceptual model
Verification & Validation
N
Y
Partitioning the model into logical processes (LPs)
Verification & Validation
N
Y
Mapping of LPs into processors
Synchronization and Networking
Programming of LPs
N
Verification & Validation
Y
Experimentation
Output analysis
Implementation and further work

Figure 6.1 - The proposed methodology for distributed enterprise simulation

6.1 Introduction

To date, much of the parallel discrete event simulation research focused on multiprocessor platforms (Carothers et al., 1997). However, Ikonen and Porras (1998) noted that in recent years the use of networked workstations for distributed applications gained more popularity. The cost involved in distributed simulation can be kept down as most of the equipment is already available. Low cost of equipment and incremental scalability are other main advantages of using a distributed system over parallel systems. Hence, the use of networks of workstations interconnected through local area network (LAN)/ wide area network (WAN) has been evolving into a popular and effective platform for distributed simulation. Idle cycles of workstations can be used to run distributed applications on networks of workstations. Moreover, a network of workstations can be considered as a parallel computer, or 'hypercomputer', whose performance is similar to that of a parallel machine but is achieved at much lower cost (Cabillic and Puaut 1997).

In distributed simulation, the simulated system is partitioned into a set of sub systems that are simulated by a set of processors that communicate by sending and receiving timestamped messages over the network (Lin, 2000). These messages are passed through the network (Figure 6.2). Carothers et al. (1997) noted that distributed simulation is one of the most demanding applications which can be run on a computer network. Moreover, workstations and the network itself are subject to heavy external loads in an open network computing environment (due to other applications executed) (Carothers et al., 1999). This leads to the degradation of performances of applications executed over the network, including simulation. However, Ikonen and Porras (1998) pointed out that disadvantages of slow communication through a network can be overcome by the proper planning of the simulation system. Distributed simulation is affected by all elements of a network system including software, hardware and communication network. Therefore, design of the network also plays a critical role in performances of a distributed simulation.

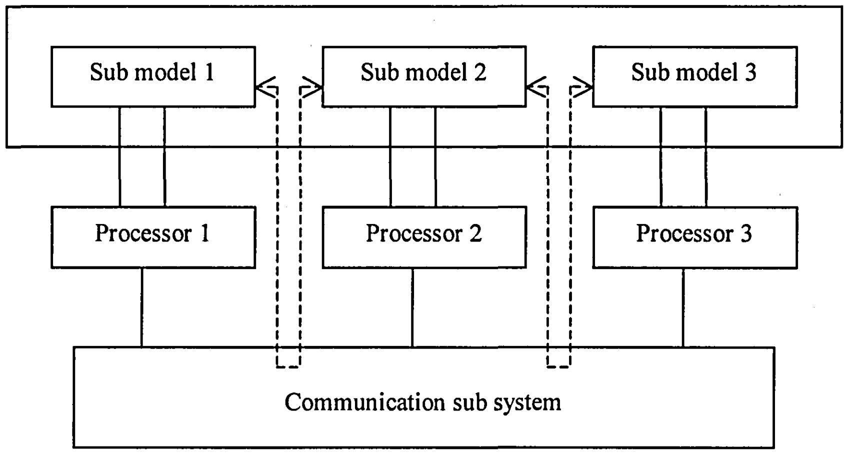


Figure 6.2 – Distributed simulation and communication network (Dado et al., 1993)

Interest in parallel and distributed simulation arose first with the problem of synchronization and it is a problem that has remained in the focus of most research in the area (Nicol and Fujimoto, 1994). Das (2000) also mentioned that most of the research in distributed simulation (also in parallel simulation) so far is centred on design of synchronization protocols and their evaluation with various simulation benchmarks.

Simulations pose unique synchronization constraints due to their underlying sense of time. When the simulation state can be simultaneously changed by different processes, actions by one process can affect actions of another (Nicol, 1993). However, the outcome of a simulation should not depend on the way it is simulated. That is, if the same model is simulated using distributed simulation and sequential simulation, users must be able to get an identical outcome. In addition, it should be repeatable. For this purpose individual simulation models need to be synchronized.

The objective of this chapter is to provide a discussion of networking issues relating to distributed simulation and present an approximate synchronization mechanism for distributed enterprise simulation. The next section presents a description on computer networks that include network topologies, communication protocols and network protocols. Section 6.3 discusses synchronization issues and different synchronization protocols. An approximate synchronization approach is presented in section 6.4. The chapter ends with a summary.

6.2 Computer networks

Based on the geographical area covered, a computer network can be either a LAN, WAN or a metropolitan area network (MAN). LAN covers a limited physical area (a building, a company, or a campus). WAN covers a wide geographical area and can even extend over countries. MAN is an intermediate between former two and can be extended to cover a city. In general, LANs provide a much "friendlier" environment for distributed simulation systems than WANs or MANs (Fujimoto, 2000).

When compared against other aspects of distributed simulation such as synchronization and partitioning, possibility of changing the networking infrastructure and networking protocols is less as it is expected to utilize existing networks to run distributed enterprise simulation. Following descriptions on network topologies and network protocols are presented in order to provide a complete set of literature on issues relating to distributed enterprise simulations.

6.2.1 Network topologies

Different types of network designs or network topologies are available for distributed computer systems. Historically LANs were based on either Bus or Ring networks (Figure 6.3). In Bus networks, all stations are connected to a single transmission path that spans the whole length of the network. In ring networks, stations are generally connected to a ring using active interfaces. It can be considered as a sequence of point-to-point links closed on itself. Recently, Star based or Tree based networks (Figure 6.4) are gaining popularity over Bus and Ring based systems. In a Star network, all stations are connected to a central node by dedicated links. Links can be established with unshielded twisted pair (UTP) cables, shielded twisted pair (STP) cables, Fibre Optics cables, wireless systems etc. The tree topology consists of a hierarchical structure, with stations being the leaves of the tree. Stations are connected to nodes at the next higher level of the structure. A recent phenomenon is the appearance of switched LANs, such as those based on Ethernet or asynchronous transfer mode (ATM) switching technology (Fujimoto, 2000). Advantages and disadvantages of these topologies were extensively discussed by Abeysundara and Kamal (1991).

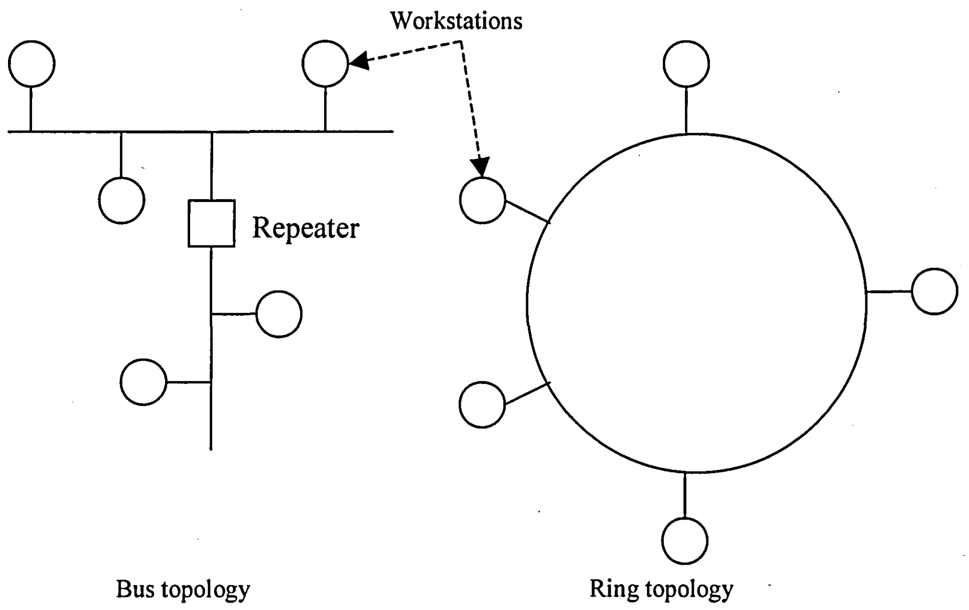


Figure 6.3 – Bus topology and Ring topology

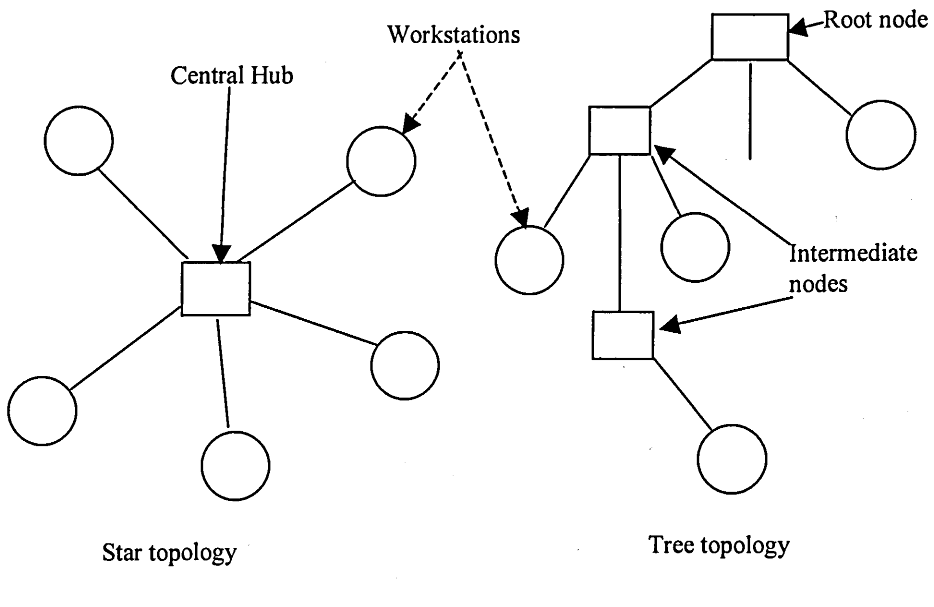


Figure 6.4 – Star topology and Tree topology

6.2.2 Network protocols

Overall performance of the network (thus of distributed simulation) depends on the type of network protocol employed (Sohl, 2002). TCP (Transmission Control protocol)/ IP (Internet Protocol), UDP (User Datagram Protocol) and IPX (Internetwork Packet Exchange)/ SPX (Sequenced Packet Exchange) are some of the network protocols that can be used in a distributed environment. Each protocol has its own merits and demerits. TCP is a connection oriented protocol. It guarantees that data packets will arrive at their destination error free and in the order in which they were sent (Kirchner, 1997). But it also incurs an overhead which leads to latencies. On the other hand, UDP is a connectionless protocol which merely sends a packet of data with no guarantee that it will arrive at its destination. IPX/SPX is a protocol for Novell based systems. Dewire (1997) provided a detailed discussion about different network protocol types, and advantages and disadvantages of them.

As noted previously, simulation developers have only limited control over networking aspects and are expected to use existing network topologies and protocols. Most of the networks that exist in small to medium size enterprises are fall into star or tree categories and it is expected to use TCP/IP protocol, which is the most commonly used protocol for the implementation of distributed simulation system.

6.3 Synchronization

State variables, a global clock and an event list do not exist in a parallel or distributed simulation system. On the other hand, individual logical processes (LPs) can be considered as sequential simulations with state variables, virtual clock and an event list (Mehl and Hammes, 1993). Bagrodia (1996) viewed distributed simulation as a collection of sequential discrete-event simulation models, which communicate each other with timestamped messages. A synchronized simulation system makes sure that each LP processes arriving messages in their timestamped order and not in their real time arriving order. This requirement is referred to as the local causality constraint (Fujimoto, 1999). To satisfy the local causality constraint, a number of synchronization algorithms have been proposed. Such algorithms can be classified into two classes as synchronous and asynchronous (Kim et al., 1997).

6.3.1 Synchronous systems

In synchronous systems, synchronization of communicating subsystems is achieved by means of a global clock whose transitions define points in the time when communication transactions can take place. All LPs must have the same simulated time under this system. Every LP must process all events in a time interval before any of the LPs are allowed to begin processing events at next time step and latter time steps. This strategy considerably simplifies the implementation of correct simulation by avoiding deadlock and need for overwhelming number of messages required by synchronization protocols in asynchronous simulation (Ferscha, 1995). The imbalance of work across LPs in certain time steps on the other hand naturally leads to idle times and represent a source of inefficiency (Ferscha and Tripathi, 1994). Also a synchronous simulation would constrain the time unit to the smallest time increment of the whole system. Pham et al., 1998 noted that in some cases it is difficult to define a global clock for a simulation.

6.3.2 Asynchronous systems

Asynchronous simulation relies on the presence of events occurring at various simulated times that do not affect each other. Concurrent processing of events effectively speeds up a simulation. Righter and Walrand (1989) mentioned that asynchronous simulation has received the greatest attention due to its potential high performance. However, asynchronous simulations are susceptible to causality errors (Ferscha and Tripathi, 1994). Numerous algorithms have been developed for synchronization of asynchronous parallel and distributed simulation in order to avoid causality errors. These algorithms are known as synchronization protocols and can be broadly classified into two categories: conservative and optimistic protocols (Fujimoto, 1999).

6.3.2.1 Conservative synchronization

Historically, first synchronization algorithms were based on conservative approaches. The idea of conservative synchronization was proposed by Chandy in

1977 and independently by Bryant. Conservative approaches strictly impose the local causality constraint and guarantee that each process only processes events in non-decreasing timestamp order (Turner, 1998). Fujimoto (1990) noted that no causality error can ever occur in an asynchronous simulation if and only if every LP processes events in non-decreasing timestamp order only. With this simple mechanism, a LP must block if it does not own any safe event to proceed. However, this algorithm does not prevent a simulation from running into a deadlock. It is possible that some LPs become blocked and each of them waits indefinitely for each other in a cyclic fashion (Vee and Hsu, 1999). Misra (1986) proposed to use null messages to avoid the deadlock. A null message with timestamp T sent from a LP is an assurance given by the LP that later it will not send a message with a timestamp smaller that T.

The null message algorithm introduced a key property called lookahead utilized by virtually all conservative synchronization algorithms (Fujimoto, 1999). Lookahead is the amount of time that a process can look into the future. If a LP is at simulation time T, and it can guarantee that any message it will send in the future will have a timestamp of at least T+L regardless of what messages it may later receive, the LP is said to have a lookahead of L. Nicol (1996) provided a discussion of different dimensions of lookahead.

Conservative algorithms can either be deadlock avoidance algorithms or deadlock detection and recovery algorithms. Although null messages are used to avoid deadlocks, they lead to increase in network traffic. Chandy and Misra (1981) introduced a deadlock detection and recovery approach. This algorithm allows processors to fall into a deadlock state, then detects the deadlock and breaks it. Since the original Chandy and Misra algorithm, a number of modified algorithms was introduced based on the conservative synchronization principle. Boukerche and Trooper (2001), Calinescu (1995), Fujimoto (1999), Nicol (1993), Reynolds (1988) and Vee and Hsu (1999) described and compared these modified algorithms.

6.3.2.2 Optimistic synchronization

Optimistic synchronization algorithms detect and recover from causality errors rather than strictly avoiding them. In contrast to conservative mechanisms, optimistic approaches need not determining when it is safe to proceed; instead, they determine when an error has occurred and invoke a procedure to recover. The best-known optimistic protocol is time warp protocol based on virtual time (simulated time) introduced by Jefferson (1985). This protocol executes every message as soon as it arrives. If a message M_t with an earlier timestamp subsequently arrives, the rolls back its state of the time M_t, and re-execute from that point. All messages sent before M_t are cancelled by sending anti-messages. To support roll back, Lin (2000) mentioned that an input queue, output queue, a local clock and a state queue should be maintained. This leads to increased usage of memory, which is a major drawback of the time warp protocol. Optimistic synchronization approaches including modified ones are described and compared by Fujimoto (1990), Fujimoto (1998), Reynolds (1988), and Vee and Hsu (1999).

6.3.2.3 Conservative vs. Optimistic synchronizations

The primary emphasis of research in distributed simulation has been on proposing and proving correctness of synchronization schemes. The most crucial question for practitioners is the choice of a synchronization protocol: i.e. conservative or optimistic, for a particular simulation problem (Ferscha et al., 2001). Both conservative and optimistic protocols have their own merits and drawbacks. The implementation of conservative algorithm is simpler than the implementation of optimistic protocols (Baukerche and Tropper, 2001). However, due to their strict adherence to local causality constrain, conservative protocols may not fully exploit the inherent parallelism of a simulation (Peterson and Willis, 1999; Porras et al., 1997). Conservative protocols are also prone to deadlock and null messages used to break the deadlock may lead to increase in network traffic resulting in latencies. Since optimistic protocols do not strictly adhere to local causality constraint, they have more potential to exploit the parallelism of a simulation. But the rollback mechanism used to overcome causality errors is often time consuming and needs to keep the state of a simulation in computer memory, resulting in increased requirements for computational and communication resources. (Calinescu, 1996; Ferscha et al., 2001). Ferscha (1995) and Fujimoto

(1998) compared advantages and disadvantages of two protocols in detail. Sanchez et al (1996) compared these protocols based on aggressiveness and risk terminology introduced by (Reynolds, 1988). Aggressiveness evaluates a protocol's ability to exploit the parallelism. Risk measures the possibility of causality violations. Conservative algorithms are non-aggressive and non-risk while optimistic algorithms are aggressive and risk protocols. It has been concluded that neither conservative nor optimistic classes of synchronization algorithms proved to be strictly better than the other (Das, 2000 and Sanchez et al., 1996). In light of this, a new class of algorithms called hybrid or adaptive protocols was introduced (Das, 2000). These protocols take an intermediate approach between purely conservative and purely optimistic approaches and contain some characteristics of both main approaches mentioned earlier. Das (1996 and 2000) and Hamnes and Tripathi (1994) described a number of adaptive protocols.

6.4. The proposed synchronization approach

Most of the distributed simulation systems developed so far are systems created for a specific situation using programming languages such as C++, Java, simula etc. Therefore it is possible to save state variables at different time points when executed. This enables implementation of the optimistic synchronization protocol, which requires rolling back to a previous simulation point of time, if the local causality constraint is violated. The simulation engine could be designed in such a way that it could predict entity creation times, processing times, delay times etc. With these it is also possible to calculate a value for lookahead that is critical for the conservative simulation protocol.

Rolling back to a previous time may not always feasible with commercial simulation software (which will be used to implement distributed enterprise simulation), as saving of state variables at different points of time can not be easily implemented. Therefore to synchronize different modules that are running in distributed simulation environment, a conservative simulation protocol was selected. If minimum processing times for distributed simulation models can be calculated, these values could be taken as lookahead values for respective simulation models. With them, a null message passing algorithm can be implemented to synchronize the distributed simulation system. However, since entity creation times, processing times, delay times etc. are generated by simulation engine of the model based on statistical distributions specified, it might not be possible to calculate a definite lookahead value for simulation modules if commercial simulation software is to be used. In addition, some applications such as distributed manufacturing may not require a strictly synchronized environment. In these situations an approximate synchronization approach could be used to synchronize a distributed enterprise simulation system as it is more simple and straightforward to implement than mechanisms that strictly synchronize the system.

6.4.1 An approximate synchronization mechanism for distributed enterprise simulations

The approximate synchronization mechanism can be implemented with an appropriate message passing mechanism that links different simulation models created using commercial simulation software. It does not attempt to execute all simulation models in a strictly synchronized environment. Instead different models are allowed to run at different but approximately close simulation times (STs) without using a lookahead. This is achieved through simulation models comparing STs of their own with STs of the other models. If the simulation time of a model is higher than any other model, the faster model pauses till slower running model reaches paused model's ST. As simulation models proceed in different time steps and due to delays take place in message passing, it is impossible to force them to run at the same simulation time.

The concept behind this mechanism is simple and could be implemented with any simulation system including systems built with most of the commercial simulation software packages. The basic steps of the mechanism for a distributed simulation with only 2 models as follows (figure 6.5).

Model X
Model Y

Figure 6.5 – Synchronization of 2 models

- Model X sends its simulation time (ST_X) to model Y
- If simulation time of model Y (ST_Y) is higher than ST_X then model Y sends back its simulation time (ST_Y) to model X and pauses the execution of the model Y.
- After (ST_Y - ST_X) simulation time model X sends a message to resume Model Y.

An identical process takes place when Model Y sends its simulation time to Model X.

However, if more than two distributed simulation models are to be used, above approach may result in generating too many messages leading to increased network traffic which may have detrimental effects on performance of the network. The number of messages passed for synchronization can be reduced by introducing an additional component (TPU – Time Processing Unit) for processing times sent by distributed simulation models (Figure 6.6).

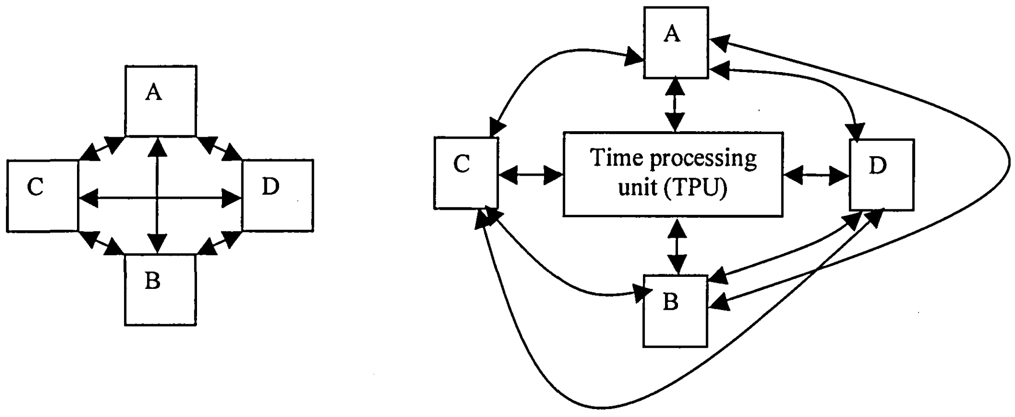


Figure 6.6 – Synchronizing mechanism without and with TPU

Instead of sending ST of one model to rest of the models all simulation models send their STs to the time processing unit (TPU). After determining the lowest ST, it pauses all simulation models except the slowest one. Before pausing, faster simulation models send their current STs to the slowest model which uses these times for scheduling the resumption of paused models. Once the ST of the slowest model reaches the ST of a paused model, it sends a message for resuming the paused model. The process continues till all the paused models are resumed. This mechanism forces distributed simulation models to run at approximately same ST. The TPU consists of 2 main elements. While the first element requests STs from distributed simulation models (figure 6.7), the second element processes the STs received from distributed simulation models (figure 6.8).

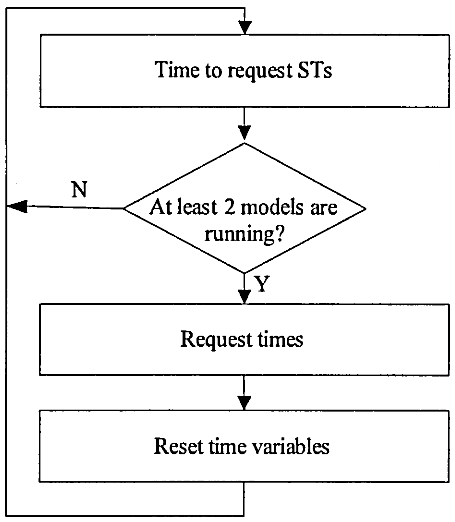


Figure 6.7 – Requesting simulation times at TPU of approximate synchronization algorithm

Figure 6.7 shows how the TPU requests STs from distributed simulation models. At predefined time intervals of the real time clock (time interval can be varied) TPU requests STs from simulation models. However, messages that request STs are passed to simulation models only if at least 2 simulation models are running in the system. If messages are sent, variables that store STs (when received from simulation models) are reset to zero.

Once a ST is received from a simulation model (responding to the ST request message from the TPU), the time processing part of the TPU updates time variables by recording the ST and the name of the simulation model from which the ST was received. It then checks whether all simulation times are received (values of time variables higher than zero) if at least 2 models are running. This is carried out in order to preventing deadlock situations as paused simulation models also send their STs to the TPU responding to the message of requesting STs. If all STs are received and at least 2 models are running, then the TPU determines the lowest ST and slowest simulation model. Then it sends messages to faster models requesting them to pause with the name of the slowest model, and updates variables by changing the status of faster models to 'Pause state' (figure 6.8).

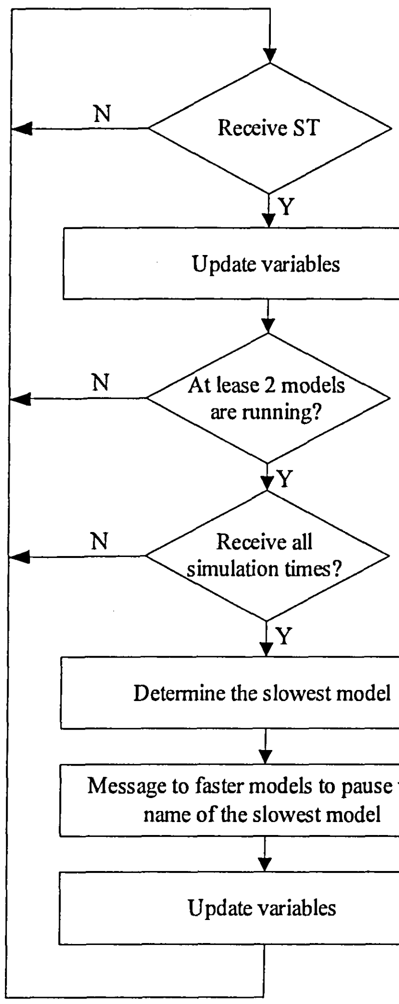

Receive ST
N
Y
Update variables
At lease 2 models are running?
N
Y
Receive all simulation times?
N
Y
Determine the slowest model
Message to faster models to pause with name of the slowest model
Update variables

Figure 6.8 – Processing simulation times at TPU of approximate synchronization algorithm

Part of the approximate synchronization mechanism is also incorporated into individual simulation models that are distributed across the network. Figures 6.11 to 6.14 show different processes of the approximate synchronization mechanism included in distributed simulation models. When a message is received from the

TPU requesting ST, individual simulation models send their STs to TPU (figure 6.9).

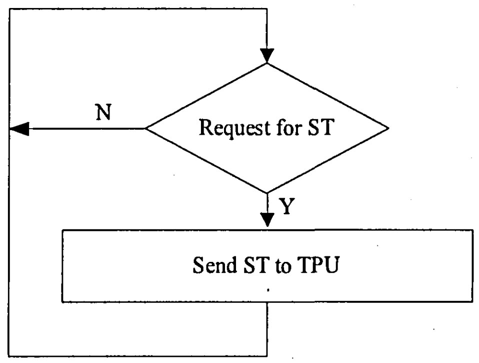


Figure 6.9 – Sending simulation time to TPU from distributed simulation models

As noted in the previous section, these times are processed at the TPU and messages are sent to faster models requesting them to pause. If a model receives a message requesting it to pause, it checks whether the model is already in pause state. If not it sends its current ST to the slowest simulation model, updates variables to indicate that it is in 'Paused state' and pauses itself (figure 6.10).

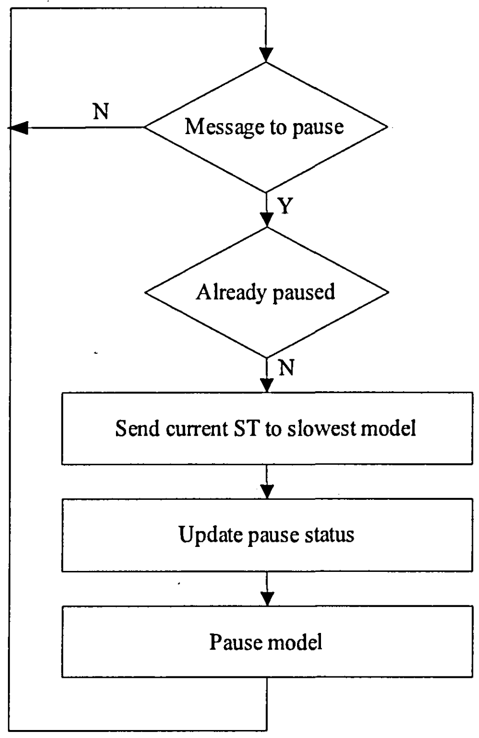


Figure 6.10 – Pausing a faster simulation model

Once the slowest model receives a ST time from a pausing model, it calculates the difference between its current ST and ST of the paused model (PTime). If the time difference is less than 0.5 STs (The value is an arbitrary figure and can be changed if necessary), a message is sent to the paused model to resume. This is carried out in order to take delays occurred in message passing into account. If the ST difference is greater than 0.5, the slowest model schedules a message to be sent for resuming the paused model after PTime (figure 6.11).

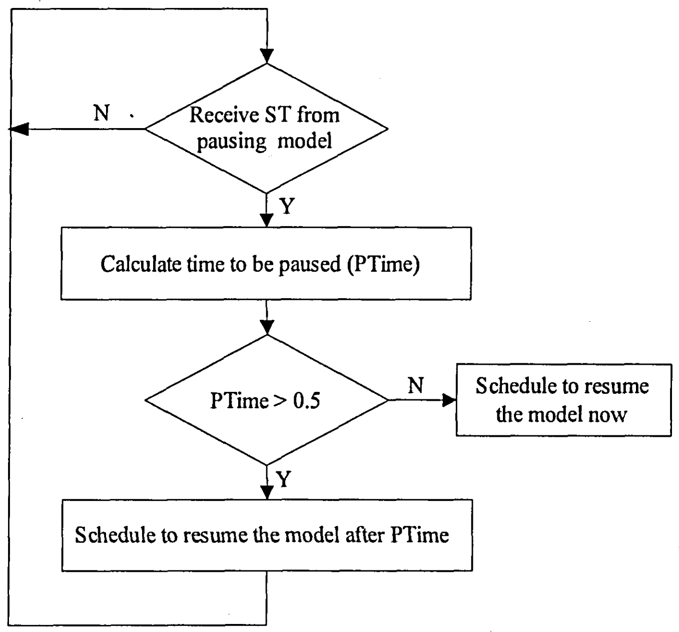

Receive ST from pausing model
N
Y
Calculate time to be paused (PTime)
PTime > 0.5
N
Schedule to resume the model now
Y
Schedule to resume the model after PTime

Figure 6.11 – Scheduling to resume a pausing model at slowest model

When a message arrives to a paused model requesting it to resume, the model updates its state variable from 'Paused state' to 'Resume state', sends a message to the TPU indicating that it resumed, and resumes itself (figure 6.12). The TPU updates the status of the resumed model when it receives such a message. (Figure 6.13)

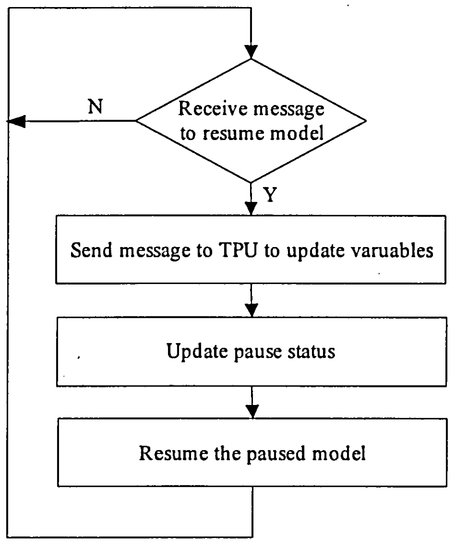

N
Receive message to resume model
Y
Send message to TPU to update varuables
Update pause status
Resume the paused model

Figure 6.12 – Resuming a paused model

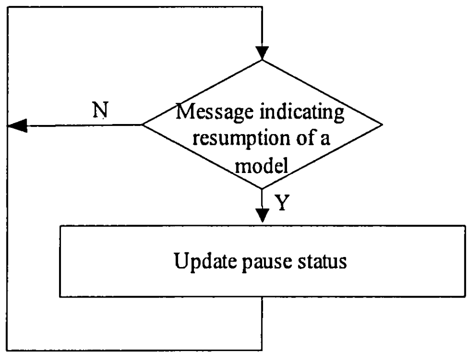


Figure 6.13 – Updating variables when resuming a model

6.4.2 Illustrating the approximate synchronization mechanism

In order to illustrate the effectiveness of the approximate synchronization mechanism, three distributed simulation models were executed without and with the synchronization mechanism. Real time (in seconds) was measured from the start of the simulation at every 10th simulation unit time for each simulation model for 500 simulation unit times. Figures 6.14 and 6.15 show graphs without synchronizing and with synchronizing respectively. The models were executed in a local area network with Tree topology which uses TCP/IP protocol. Windows XP Professional was the operating system of the one workstation and Windows 2000 professional was the operating system of the other two workstations. MSMQ

2.0 was used as the message passing middleware. MSMQ was employed in workgroup mode without a MSMQ server. A detailed discussion of MSMQ and other message passing middleware will be presented in the next chapter (chapter 7)

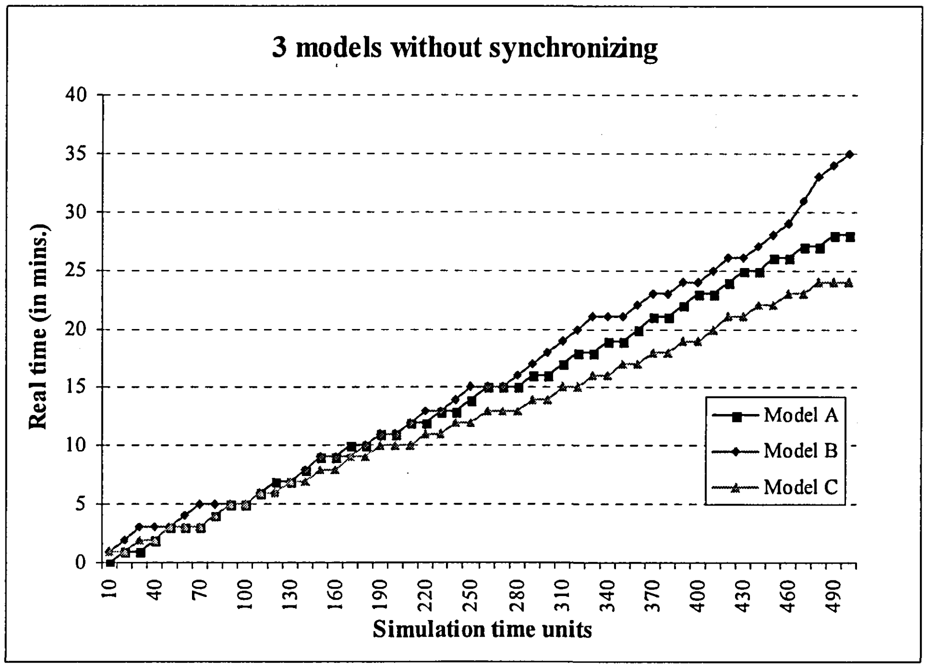


Figure 6.14 – Execution of models without synchronization mechanism

The figure 6.14 shows that three models are running at different simulation times. At 400th simulation time model C was the fastest and model B was the slowest. If a messages passed from A and B to C, they may not satisfy the local causality constraint.

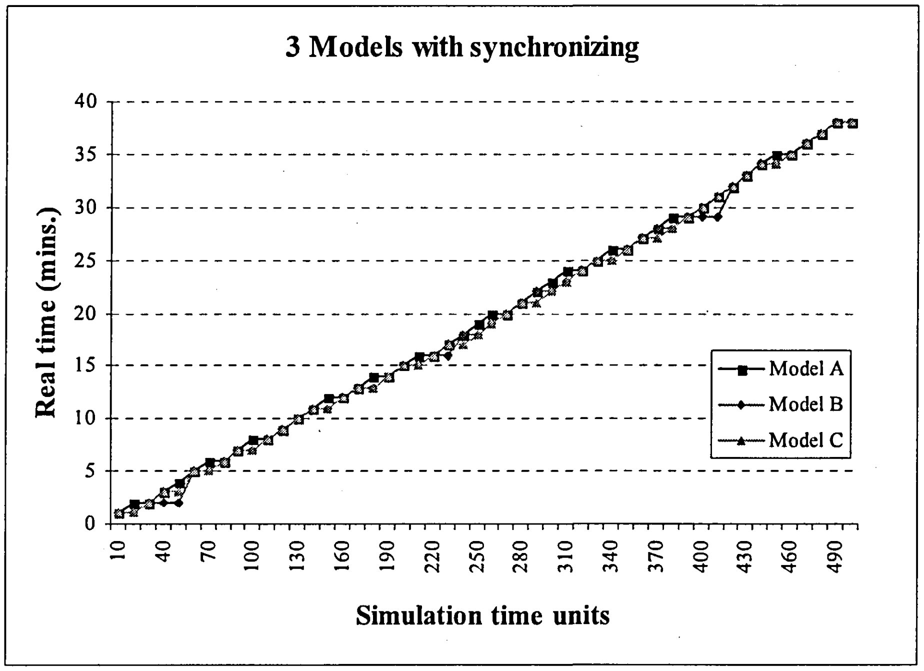


Figure 6.15 – Execution of models with synchronization mechanism

Figure 6.15 shows that all three models are running at approximately same simulation time thus avoiding the occurrence of the local causality constraint.

Summary

The chapter presented an approximate synchronization mechanism for distributed enterprise simulations. In addition, it also briefly described networking and synchronization issues relating to distributed simulation. According to the proposed methodology for distributed enterprise simulation (figure 6.1), the partitioned logical processes can be transformed into computer simulation models and executed in a distributed simulation environment. The next chapter illustrates the implementation of distributed enterprise simulation with a hypothetical case study which focuses on distributed manufacturing applications.

Chapter 7
Implementation of the distributed enterprise simulation

Chapter 6 described networking and synchronization issues relating to distributed enterprise simulation. It also proposed an approximate synchronization approach, which is particularly suitable for distributed manufacturing applications. This chapter presents a detailed approach for implementation of distributed enterprise simulation (see figure 7.1). A brief discussion of middleware is also included as middleware is used for message passing to synchronize and pass parameters between simulation models distributed across the network. In order to simplify the implementation process and to reduce the time and cost involved, it was decided to use commercial simulation software, and widely available and cost effective technologies to implement the distributed enterprise simulation. A hypothetical case study focused on distributed manufacturing is used to illustrate the proposed approach for implementation. Arena, MSMQ and VBA used as the commercial simulation software, middleware and application program interface (API) respectively for implementing the case study presented.

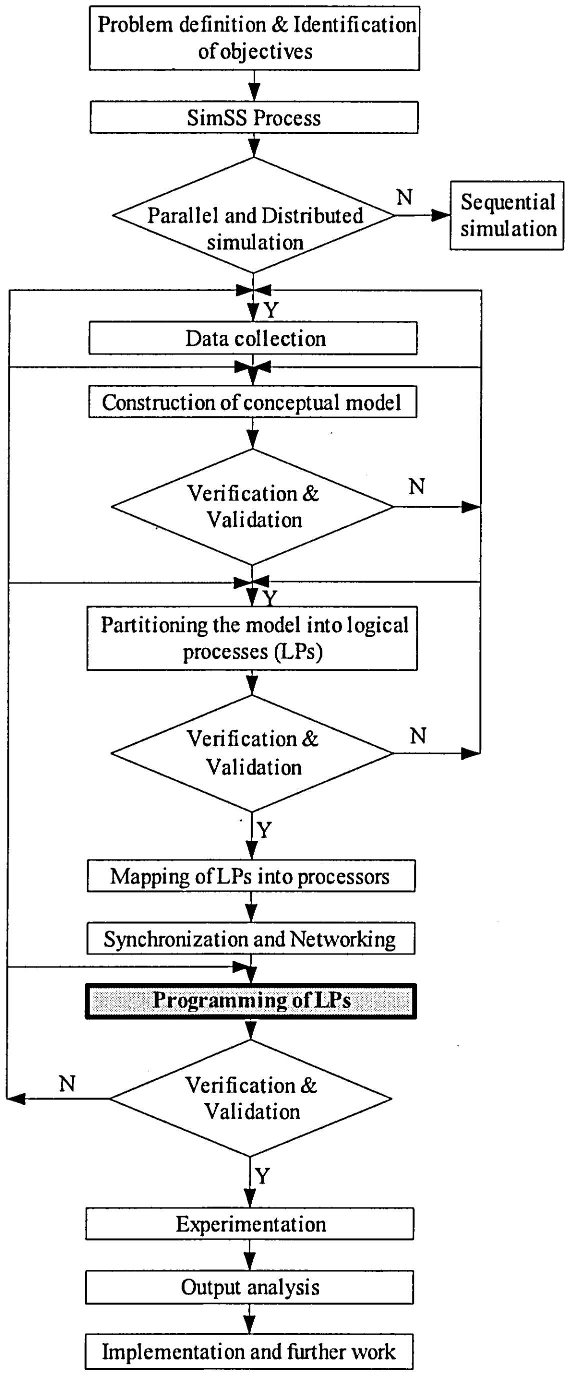

Problem definition & Identification of objectives
SimSS Process
Parallel and Distributed simulation
N
Sequential simulation
Y
Data collection
Construction of conceptual model
Verification & Validation
N
Y
Partitioning the model into logical processes (LPs)
Verification & Validation
N
Y
Mapping of LPs into processors
Synchronization and Networking
Programming of LPs
N
Verification & Validation
Y
Experimentation
Output analysis
Implementation and further work

Figure 7.1 - The proposed methodology for distributed simulation

7.1 Introduction

During the development of a distributed simulation, issues such as model partitioning, mapping, synchronization, and implementation (mainly the technologies and software used) need to be addressed. As noted in previous chapters, the new methodology proposes to pursue a different approach in developing and implementing distributed enterprise simulations. Generally distributed simulations are implemented using either a simulation language such as Simula or general purpose languages such as C++ and Java. This calls for not only expertise in distributed simulation, but also expertise in programming. Moreover, the need for middleware to communicate between distributed models further complicates the implementation process. A number of authors including Ikonen and Porras (1998) and Pancake (1996) criticized the implementation process of parallel and distributed simulations as time consuming, effort intensive, complex and involving steep learning curves. As a result, distributed simulations (along with parallel simulations) are still being utilized primarily in the research community, with only a limited penetration in the commercial modelling and sequential simulation community (Bass, 1999; Cai and Teo, 1999; Page, 1999; Taylor, 1998).

The new methodology for distributed enterprise simulation uses commercial simulation software to develop distributed simulation models, and uses a cost effective and simplified approach to implement distributed enterprise simulation. To illustrate the implementation process a hypothetical case study in distributed manufacturing was developed. Distributed manufacturing applications can be easily implemented with the proposed methodology, since they are not as complicated as logic circuits, telecommunication system etc., and generally do not require to be strictly synchronized.

In order to synchronize and pass parameters between simulation models distributed across a network, simulation models need to communicate with each other. Communication methods provided by operating systems often require complex programming. In a distributed simulation, middleware provides a simple and reliable solution for this problem. Middleware is a piece of software that interacts between different programs distributed across a network. It provides a higher level building block for programming than connection methods provided by operating systems by allowing applications to locate transparently across the network, providing interactions with another application or service, be independent from network service, be reliable and available, and scale up in capacity without losing function (Schreiber, 1995).

The objective of this chapter is to present a simplified and cost effective approach to implement a distributed enterprise simulation. The next section provides a short introduction to distributed manufacturing on which the implementation approach was based. Sections 7.3, 7.4 and 7.5 respectively provide descriptions on middleware, simulation software and API used for the implementation. The hypothetical case study on distributed manufacturing is presented in section 7.6 to illustrate the implementation of distributed enterprise simulation. The detailed implementation approach is shown in section 7.7. Section 7.8 briefly discusses the output from a distributed enterprise simulation as (unlike traditional sequential simulation) more than one model can generate output. Validation issues of distributed enterprise simulation is briefly presented in section 7.9. The chapter ends with a summary.

7.2 Distributed manufacturing

Confronted with growing competition, the evolution of new markets and increasingly complex global and political scenarios, today's manufacturing organizations are forced to rethink about how they are organized and operated. Not only to gain a competitive advantage over their competitors but often merely to survive, companies are now looking for innovative ways of responding to market changes, produce better quality products in more cost effective manner, manage product life cycles effectively etc. As a result, enterprises are moving towards more open architectures for integrating their activities with those of their suppliers, customers and partners within wide supply chain networks (Shen and Norrie, 1998). In manufacturing, companies may form strategic partnerships by outsourcing some of their operational activities, sharing resources or joint development of products and services etc., leading to formation of virtual manufacturing enterprises which operate in distributed manufacturing environment.

Due to their nature and also to the environment they operate, distributed manufacturing enterprises (DMEs) are highly complex and heterogeneous. The traditional manufacturing control systems have low capacity to adapt and to react to the complex and dynamic nature of DME. Therefore, attempts have made to develop distributed manufacturing architectures that can deal with complex and dynamic systems. New control and organizational architectures such as Agile, Fractal, Bionic, Random, and Holonic manufacturing architectures have been introduced over the last few years (Kadar et al., 1998; Leitao and Resviti, 2000 and 2001; Saad, 2003).

DMEs which are also known as virtual manufacturing enterprises operate in geographically distributed environment and connected together with modern communication technologies. Virtual manufacturing enterprises are ephemeral organizations in which several companies collaborate to produce a single product or product line (Venkateswaran et al., 2001). Participating in this type of collaboration allows partner organizations to use their knowledge, resources and in particular manufacturing expertise to take advantage of new business opportunities and/or gain a competitive advantage that are on a larger scale than an individual partner could handle alone. Generally these types of enterprises are established without making a long term commitment to other partners and individual partners may also carry out their own manufacturing activities independent of activities relating to the DME. To facilitate the creation of virtual manufacturing enterprises, potential partners must be quickly able to evaluate whether it will be profitable for them to participate in the proposed enterprise. Simulation provides a capability to conduct experiments rapidly to predict and evaluate the results of manufacturing decisions (McLean and Leong, 2001).

Simulation is not a strange tool for decision making in manufacturing. Law and McComas (1999) pointed out that manufacturing is one of the largest application areas of simulation, with the first uses dating back to at least early 1960s. However, traditional sequential simulation alone may not be sufficient to simulate these highly complex DME. In such situations, distributed simulation provides a promising alternative to construct cross enterprise simulations. Each partner can simulate its operation to make sure that it has the capability to perform its individual function in the DME. Later these simulation models can be integrated into a distributed simulation for simulating the whole enterprise in order to evaluate the feasibility and profitability of the proposed partnership. Use of distributed simulation allows each partner to hide any proprietary information, simulate multiple manufacturing systems at different degrees of abstraction levels, link simulation models built using different simulation software, to take advantage of additional computing power, simultaneous access to executing simulation models for users in different locations, reuse of existing simulation models with little modifications etc. (Gan et al., 2000; McLean and Riddick, 2000; Taylor et al., 2001; Venkateswaran et al., 2001). However, Peng and Chen (1996) noted that as a technique, parallel and distributed simulation is not successful in manufacturing. Most of the simulations for DMEs implemented so far are purpose build simulators created using programming languages such as C++ or Java, and with high end workstations. Furthermore, as noted previously distributed simulation itself involves long development time, cost, steep learning curves, and is often complex to manage resulting low penetration into industrial applications.

7.3 Middleware

The most important role in the networking subsystem of a distributed simulation is the efficient exchange of messages (Sohl, 2002). Message passing can be point-to-point, broadcast or multicast. The point-to-point method requires the source to pass messages directly to the destination. In broadcasting, the source sends message to all hosts, which are 'listening'. This eliminates the repeated and multiple connects needed by the point-to-point method. However, this may lead to an increase in network traffic resulting latencies. Multicast is an improvement to broadcast. It enables the source to pass messages to desired hosts.

Different techniques are used to communicate between simulation sub models. HLA (High Level Architecture) uses RTI (Run Time Infrastructure) (Buss and Jackson, 1998). Middleware such as CORBA (Common Object Request Broker Agent), GRIDS (Generic Runtime Infrastructure for Distributed Simulation) (Sudra et al., 2000), CDNS (Collaborative Distributed network Systems) can be used to pass messages between simulation models distributed across a network.

Middleware is a class of software designed to help managing the complexity and heterogeneity inherent in distributed systems. It consists of a set of enabling services which allow multiple processes running on one or more computers to interact across a network. This technology evolved during 1990s to provide for interoperability in support of the move to client/ server architecture (Bray, 2003). Bakken (2003) defined middleware as a layer of software above the operating system but below the application program that provides a common programming abstraction across a distributed system (see figure 7.2).

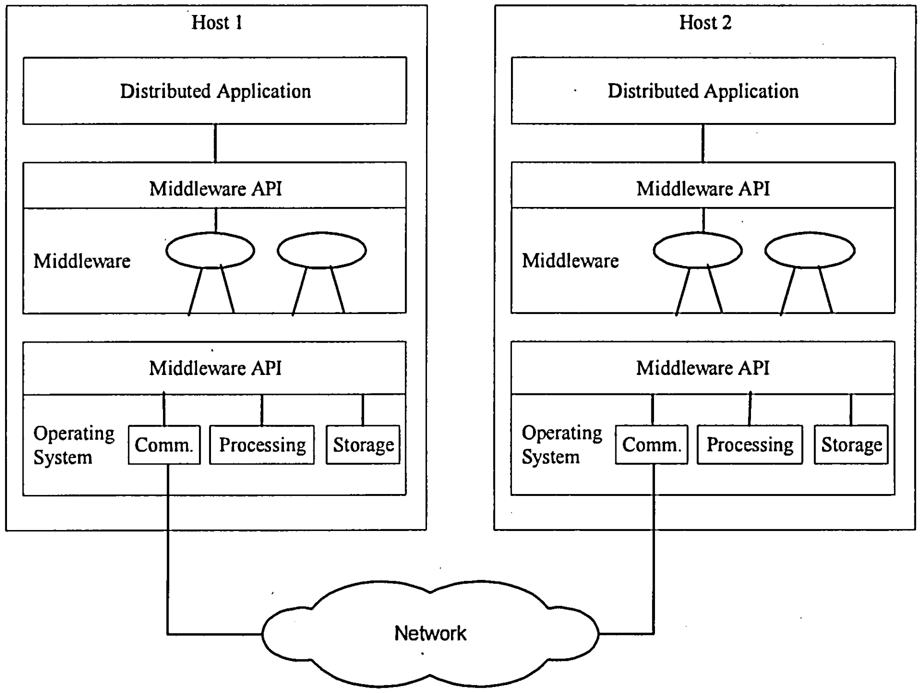


Figure 7.2 – Middleware layer in context (Bakken, 2003)

Based on programming abstractions and the kinds of heterogeneity provided beyond network and hardware, middleware can be categorized into few different forms (Bakken, 2003; Berson, 1996; Dewire, 1997).

7.3.1 Forms of middleware

Remote Procedure Calls (RPC)

A remote procedure call is a mechanism by which one process can execute another process (subroutine) residing on another, usually a remote system possibly running a different operating system. It extends the procedure call interface to offer the abstraction of being able to invoke a procedure whose body is across the network. RPC systems are usually synchronous, and thus offer no potential for parallelism without using multiple threads. They typically have limited exception handling facilities.

Message Oriented Middleware (MOM)

Message oriented middleware offers program to program data exchange, enabling the creation of distributed applications. It provides the abstraction of a message queue that can be accessed across a network. MOM is analogous to email in the sense that it is asynchronous and requires the recipients of messages to interpret their meanings and to take appropriate actions. It is very flexible in how it can be configured with topology of programs that deposit and withdraw messages from a given queue.

Object Request Brokers (ORB)

This type of middleware enables the objects that comprise an application to be distributed and shared across heterogeneous networks. It provides the abstraction of an object that is remote yet whose methods can be invoked just like those of an object in the same address space as the caller.

In addition, inter process communication (IPC) and transaction processing (TP) also can be employed to communicate between remote processes. Based on the middleware forms described above, a number of middleware architectures introduced over past few years. Followings are the widely publicized architectures summarised by Bray (2003):

7.3.2 Well known middleware architectures

Distributed Computing environment (DCE)

Developed and maintained by the Open Systems Foundation (OSF), the Distributed Computing Environment (DCE) is an integrated distributed environment which incorporates technology from industry. The DCE is a set of integrated system services that provide an interoperable and flexible distributed environment with the primary goal of solving interoperability problems in heterogeneous, networked environments. OSF provides a reference implementation (source code) on which all DCE products are based. For physical data exchange and communication DCE uses RPC.

The standard interfaces used by the DCE as well as all the source code itself, are defined only in the C programming language. Vondrak and beach (2003) noted that original DEC products were "developer's kits" that were not robust, did not contain the entire set of DCE features (all lacked distributed file services), and were suited mostly for UNIX platforms. Johnson (1991) provides more detail on DEC.

Common Object Request Broker Architecture (CORBA)

Common object request broker architecture is a specification of a standard architecture for object request brokers (ORB). CORBA specification was developed by Object Management Group, an industry group with over six hundred member companies. The ORB handles the interacting objects by behaving as an extensive object oriented RPC application program interface (API). Using CORBA compliant ORB, a process can transparently invoke a method on another object, which can be on the same machine or across a network. The process does not need to be aware of where the object is located, its programming language, its operating system or any other aspects that are not part of an object's interface. CORBA interfaces are developed with Interface Definition language (IDL) which is similar to C++. A number of authors including Minton (2003), OMG (2002) and Wallanau (1997) presented more details on CORBA.

Distributed Component Object Model (DCOM)

Distributed component object model (DCOM) is an extension to component object model (COM) that allows network based component interaction. COM refers to both a specification and implementation developed by Microsoft. With DCOM, components operating on a variety of platforms can interact as long as DCOM is available within the environment. Its distributed object abstraction is augmented by other Microsoft technologies including Microsoft Transaction Server (MST) and Active directory. Similar to DCE and CORBA, communication between different objects under DCOM are also based on RPC. COM+ which is the evolution of COM integrates MST services and message queuing into COM, and makes COM programming easier through a closer integration with Microsoft programming languages such as Visual Basic, Visual C++, and Visual J++.

Javasoft's Java/ Remote method Invocation (Java/ RMI) too is a middleware architecture based on RPC.

As it was noted DCE, CORBA, DCOM and Java/RMI are based on RPC form of middleware which mainly support synchronous communication between remote processes. DCE and CORBA are relatively more matured than DCOM and enjoy more acceptability as middleware specifications for them are published by industry groups. Furthermore, these architectures support most of the computing platforms. On the other hand, DCOM based technologies have the ability of evolving faster than DCE and CORBA as one vendor developing its own proprietary specification. However, still Microsoft Windows (including both desktop and server) is the only platform which is fully supported by DCOM.

High Level Architecture (HLA)

HLA is a standard framework that supports simulations composed of different simulation components thus encouraging reusability and interoperability. IEEE 1516 standards specify HLA as a standard for distributed simulation. For (most of the) military applications in distributed simulation HLA has been accepted as the standard architecture. However, Strassburger in Taylor et al. (2002) noted that HLA as an IEEE standard failed to gain acceptance from non-military users mainly due to its relatively high complexity.

The main requirement for middleware in the proposed distributed simulation approach is passing messages between distributed simulation models for synchronizing the distributed simulation system and for passing parameters. While, RPC based middleware can be employed to communicate between distributed simulation models, message passing based on MOM may simplify the programming task. Unlike RPC, MOM does not require a synchronous connection between remote models and is more flexible than the former. Furthermore, MOM is well suited for event driven applications.

Microsoft Message Queue (MSMQ) was selected as the middleware for communicating between distributed simulation models. Although MSMQ is a part of Microsoft DCOM/COM+ architecture, it is a MOM. Since MSMQ 2.0 is integrated into Windows 2000 (both server and professional versions) and Windows XP (as version 3.0) and available as an additional component for Windows NT, 95 and 98, it provides an cost effective solution for message passing. Application program Interface (API) for MSMQ can be developed with Visual Basic, C++ or Visual Basic for Applications (VBA). VBA is also integrated into many Microsoft applications such as MS Office, Visio etc. and a number of third party applications including Arena simulation software.

MSMQ workgroup mode can be implemented without MSMQ server mode and does not uses directory services offered by Windows 2000 server. However, if messages are required routing to a workstation in a different domain, then services of MSMQ server mode is required. The advantage of workgroup mode is that it can be deployed on a Novell environment although distributed simulation needs to be restricted to a single domain. MSMQ 3.0 which is integrated into Windows XP supports message passing with HTTP protocol. Therefore with MSMQ 3.0 messages can be passed simulation models distributed across the internet. It also supports multicasting of messages in addition to unicasting.

Applications developed with MSMQ could communicate across heterogeneous networks and with computers that may be offline. It provides guaranteed message delivery, efficient routing, security, transactional support, and priority based messaging and could operate in either domain or workgroup environment (Chapell, 1998). In a message queuing system, applications send and receive messages to message queues, which could be located in either a local or a remote computer. Applications interact with MSMQ via an API. The API developed for MSMQ could send messages containing parameters obtained from simulation model to a queue in the same computer or directly to another remote computer. API that resides in the remote computer extracts these messages from the queue and passes parameters to the simulation model.

The last chapter presented a brief discussion of network topologies and networking protocols. As noted in the same chapter, the likelihood of changing the existing network infrastructure for distribution simulation is low. To emulate the networking environment in which generally distributed enterprise simulations are expected to be implemented, it was decided to use School of Engineering's (of Sheffield Hallam University) main network to illustrate the implementation of the case study. The school's main network which is in the form of tree topology and operates in Novell NetWare environment with TCP/IP protocol. Therefore, MSMQ workgroup mode was used to implement the hypothetical case study.

7.4 Simulation software

As noted in chapter 3, there is a growing trend towards using commercial simulation software packages to implement distributed simulations. Arena simulation software was used to illustrate the implementation of the case study. However, other commercial simulation software such as Automod, Promodel, Witness etc. can also be used for this purpose. Arena is one of the popular simulation software packages used in sequential simulation. Takus and Profozich (1997) noted that it is a flexible and powerful tool that allows an analyst to capture the dynamics of a system and create animated simulation models. A number of authors including Linn et al., 2002; Venkateswaran et al., 2001 have employed Arena to implement distributed simulations. Furthermore, a recent survey carried out at Sheffield Hallam University revealed that Arena as the most widely used simulation software in both academic and industrial communities (Yapa, 2003).

7.5 Application Program Interface (API)

The API acts as the interface between simulation software and MSMQ. It extracts messages that arrive to message queues and pass parameters to the simulation model, and obtain parameters from simulation model and pass them as a message to a queue in another workstation which is part of the distributed simulation system (figure 7.3). The API for Arena and MSMQ can be written in both Visual basic for applications (VBA) and C++. Since programming of Arena with VBA is more straightforward, it was decided to use VBA instead of C++. VBA also offers a programming environment similar to popular Visual Basic programming language and, user friendly and easy to learn.

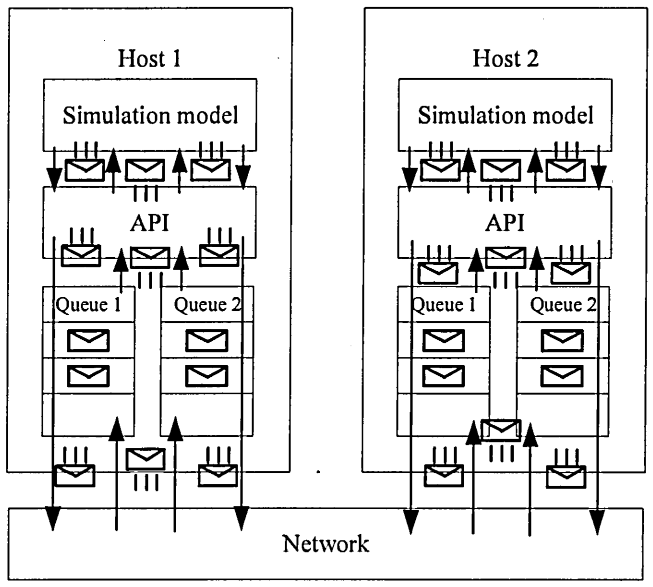


Figure 7.3 – MSMQ, API and Simulation model

In order to demonstrate the proposed implementation approach, a hypothetical case study was used. Passing product information from one model to another, synchronization, implementation of the distributed enterprise simulation using commercial software were the main point to be illustrated. These can be done either using a real world case or a hypothetical one. Due to time restrictions involved in the research, it was decided to use a hypothetical case study. The main differences between hypothetical and real case studies include number of product/ parts produced, number of firms involved in the enterprise, process flows for different products. However, these issues may not affect what is expected to demonstrate through the hypothetical case study as number of products, parts, process flows can be incorporated into the system by modifying individual simulations models. Number of partners in the enterprise can be changed by adding or removing simulation models to or from the distributed enterprise simulation with slight modifications to other models. Generic names were used for processes and work centres as the case is a hypothetical one. One of the benefits of the proposed implementation approach is ability to reuse of simulation models already developed using commercial simulation software. To highlight this point, the case assumes that already developed simulation models were modified to develop the distributed manufacturing simulation system.

7.6 Hypothetical case study

Three manufacturing firms namely A, B and C are evaluating the feasibility of forming a distributed manufacturing enterprise in order to introduce a high tech product called XYZ which potentially has a huge demand in the market. It has been recognized that individual firms can not alone produce the product as manufacturing process requires highly sophisticated equipment and complicated production processes (figure 7.4).

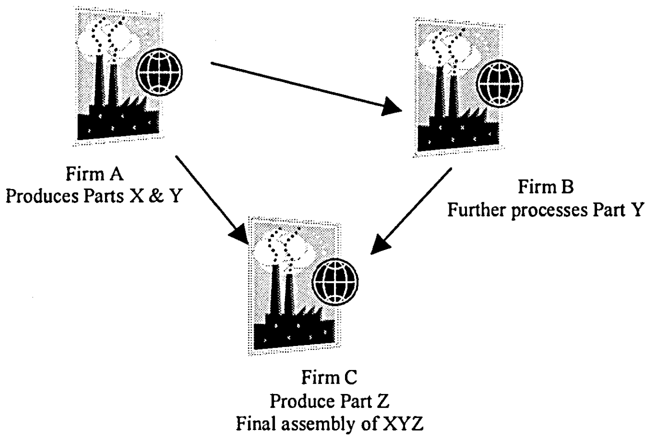


Figure 7.4 - Proposed distributed manufacturing enterprise

It was agreed that firm A which has more excess capacity is to produce and process parts X and Y. Once parts X and Y are processed at firm A, part Y to be sent to firm B which uses its patented treatment processes to further process it and part X to be sent to firm C. Part Y also sent to C after processing at firm B. At firm C part Z is to be produced and, both parts X and Y are further processed and assembled together to form product XYX. Parts are transferred in batches of 100s and transfer time is 10 hours.

Production facilities of firm A consist of 5 work centres (WCA1 to WCA5). Each Work centre contains between 3 to 4 work cells and each work cell is equipped with a number of identical machines. Parts are routed through all work cells in the same sequence if they arrive at a work centre. However, parts are not required to be processed at all work centres. In addition to parts X and Y, firm A also produces and processes parts P and Q in order to produce Product PQ. Processing sequences and processing times for parts and semi-finished products are given in figure 7.5 and table 7.1 respectively.

At work centre 4 Parts P and Q are assembled together to produce PQ and routed to Work centre 5 for finishing.

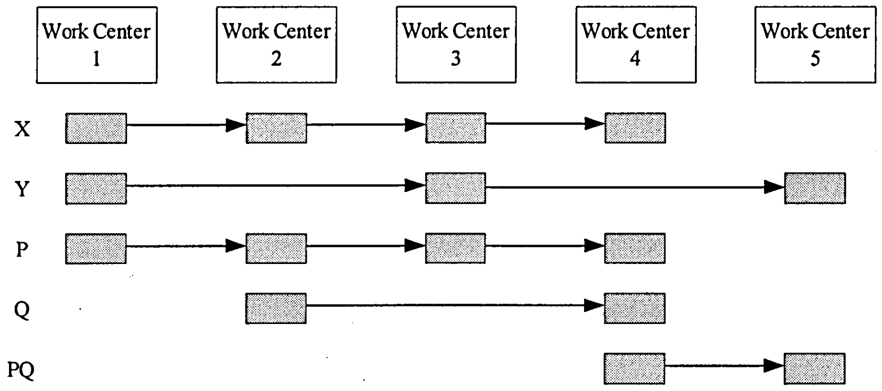


Figure 7.5 – Processing sequences at Firm A

Syntax used for processing times:

NORM( ) : Normal distribution

UNIF( ) : Uniform distribution

TRIA( ) : Triangular distribution

NORM( ) : Normal distribution

	Work Centre 1	Work Centre 2	Work Centre 3	Work centre 4	Work Centre 5
Part X	TRIA(0.5,1,1.5)	NORM(1,0.2)	UNIF(0.5,1)	TRIA(0.4,1,1.2)	
Part Y	NORM(1,0.2)		TRIA(0.5,1,1.5)		TRIA(0.5,1,1.5)
Part P	NORM(1.2,0.4)	TRIA(0.6,1,1.7)	UNIF(0.8,1.2)	TRIA(0.5,1,1.5)	
Part Q		TRIA(0.7,1,1.8)		UNIF(0.5,1)	
Product PQ				0	TRIA(0.5,1,1.5)

Table 7.1 - Processing times at Firm A

Six work centres are included in firm B's production facilities (WCB1 to WCB6). Each centre consists of a number of machines, a chemical bath and an oven. When arrive at a work centre each part or semi finished components are required to be processed at all machines, dipped in the chemical bath and baked in the oven. In addition to processing of part Y, firm B also produces product RS by processing and assembling parts R and S.

At work centre 5 Parts R and S are assembled together to produce RS and routed to Work centre 6 for finishing.

Processing sequences and processing times for parts and semi-finished products are given in figure 7.6 and table 7.2 respectively.

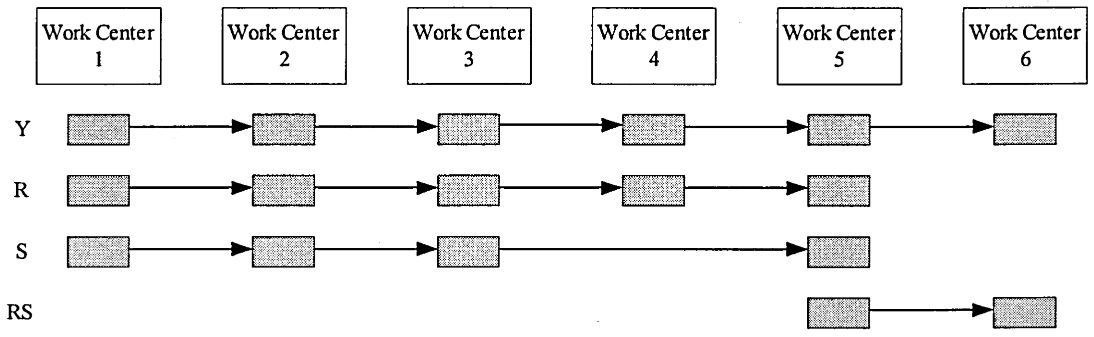


Figure 7.6 – Processing sequences at Firm B

	Work Centre 1	Work Centre 2	Work Centre 3	Work centre 4	Work Centre 5	Work Centre 6
Part Y	NORM(1,0.2)	UNIF(0.8,1.2)	NORM(1.2,0.3)	NORM(1,0.2)	UNIF(0.8,1.2)	TRIA(0.5,1,1.5)
Part R	NORM(1.1,0.2)	UNIF(0.5,1)	UNIF(0.7,1.3)	NORM(1.1,0.4)	NORM(1.0,0.2)	
Part S	UNIF(0.5,1)	NORM(1.1,0.2)	NORM(1.1,0.2)		NORM(1.1,0.2)	
Product RS					0	TRIA(0.5,1,1.5)

Table 7.2 - Processing times at Firm B

As at firm B, production facilities of firm C consists of 6 work centres (WCC1 to WCC6). Each centre contains 2 work cells with 4 identical machines. If a part or semi-finished product comes to a work centre, it needs to be routed through both work cells. In addition to processing of part X, Y and Z, and assembling product XYZ, firm C also produces product TU by processing and assembling parts T and U.

At work centre 4 Parts T and U are assembled together to produce TU then sent to work centres 5 and 6 for further processing and finishing respectively. At work centre 5 parts X, Y and Z are assembled together to make product XYZ. Both TU and XYZ are sent to Work centre 6 for finishing.

Processing sequences and processing times for parts and semi-finished products are given in figure 7.7 and table 7.3 respectively.

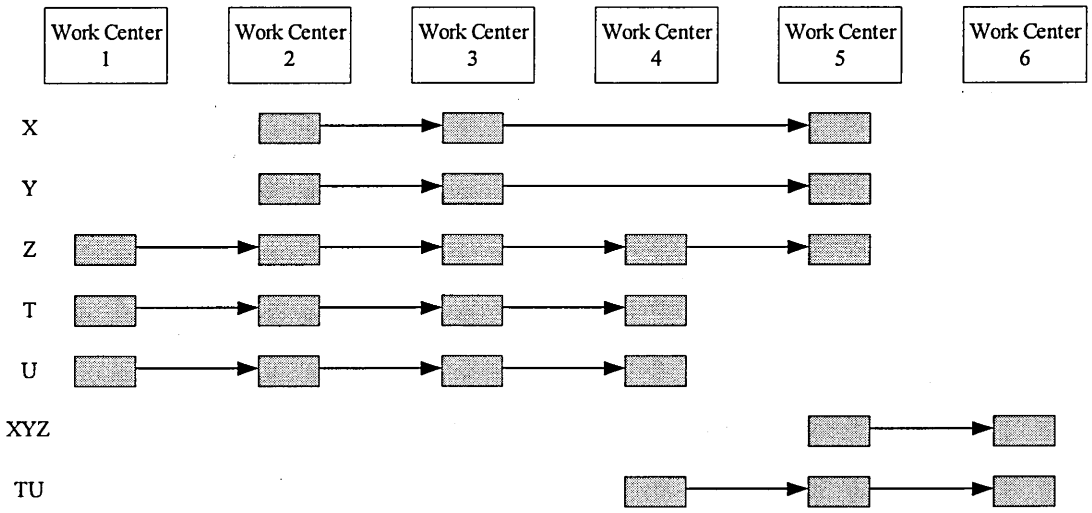


Figure 7.7 – Processing sequences at Firm C

	Work Centre 1	Work Centre 2	Work Centre 3	Work centre 4	Work Centre 5	Work Centre 6
Part X		NORM(1,0.2)	UNIF(0.8,1.2)		NORM(1,0.2)	
Part Y		UNIF(0.8,1.2)	NORM(1,0.2)		UNIF(0.8,1.2)	
Part Z	NORM(1,0.2)	NORM(1,0.2)	UNIF(0.8,1.2)	NORM(1,0.2)	UNIF(0.8,1.2)	
Part T	UNIF(0.5,1.4)	UNIF(0.5,1.2)	NORM(1.1,0.1)	NORM(1,0.2)		
Part U	NORM(1.1,0.1)	NORM(1,0.2)	UNIF(0.5,1.4)	UNIF(0.5,1.2)		
Product XYZ					0	TRIA(0.5,1,1.5)
Product TU				0		TRIA(0.5,1,1.5)

Table 7.3 - Processing times at Firm C

It was revealed that all 3 firms have been using simulation previously for analyzing their production systems and have already built simulation models using Arena simulation software for their production facilities. Furthermore, all firms are reluctant to pass their information about manufacturing processes to other firms or third parties. Therefore it was agreed to use existing simulation models, modify them to include activities of the proposed enterprise and execute in distributed simulation environment.

7. 7 Implementation

IDEF0 conceptual models were developed for the proposed distributed manufacturing enterprise and individual firms in order to reflect their activities relating to the enterprise (figure 7.8) and independent activities of individual partners (figures 7.9 7.10 and 7.11). In order to simplify the illustration, the proposed enterprise was decomposed only up to the level of work centres.

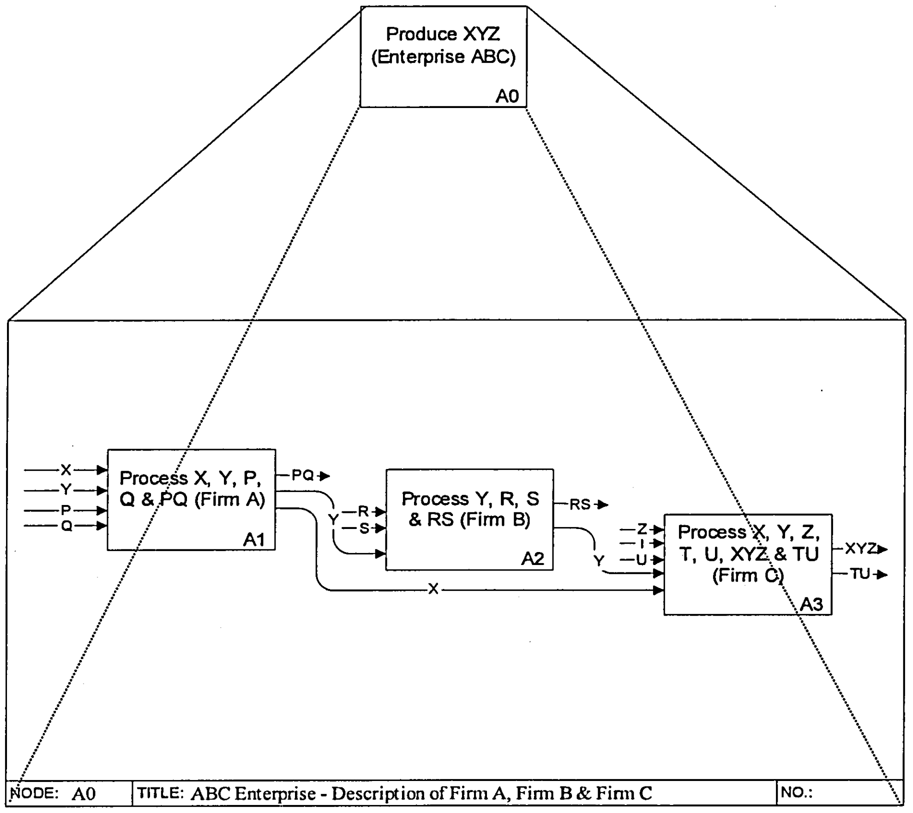

Produce XYZ (Enterprise ABC)
A0
X
Y
P
Q
Process X, Y, P, Q & PQ (Firm A)
A1
PQ
Y
R
S
Process Y, R, S & RS (Firm B)
A2
RS
Z
I
U
Y
X
Process X, Y, Z, T, U, XYZ & TU (Firm C)
A3
XYZ
TU
NODE: A0
TITLE: ABC Enterprise - Description of Firm A, Firm B & Firm C
NO.:

Figure 7.8 – Distributed manufacturing enterprise

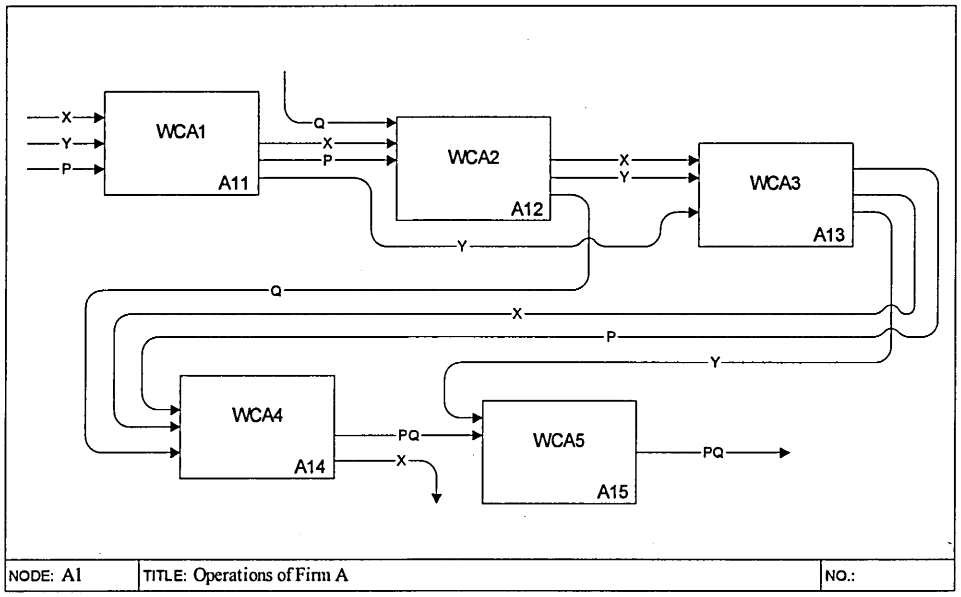


Figure 7.9 – Manufacturing operations of Firm A

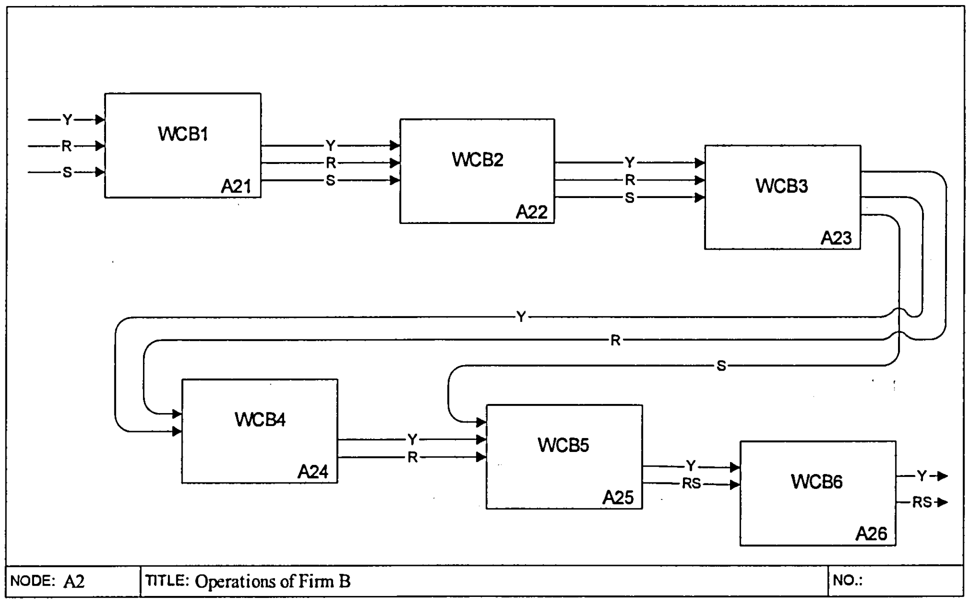


Figure 7.10 – Manufacturing operations of Firm B

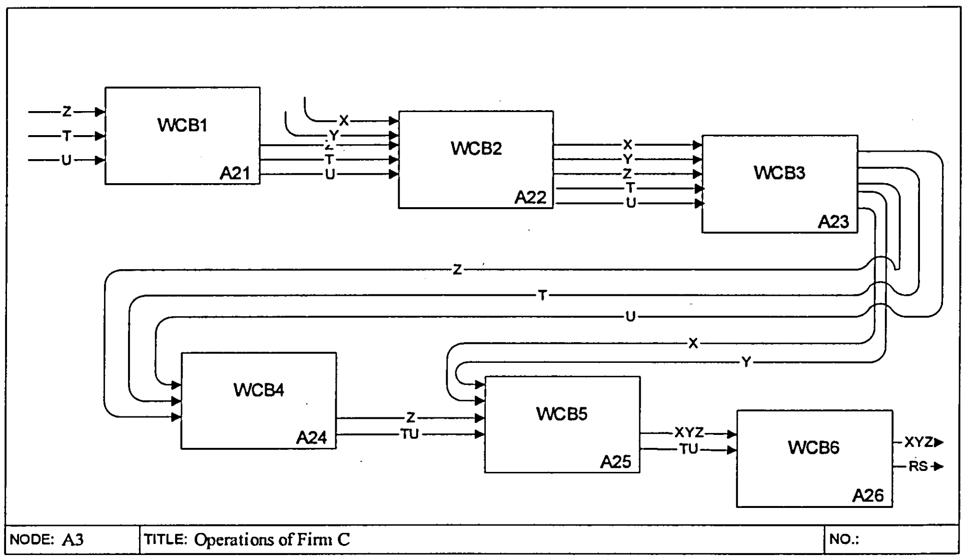


Figure 7.11 – Manufacturing operations of Firm C

Arena simulation models already developed by 3 firms (in order to illustrate the reusability of existing simulation models) were modified to accommodate operations of the proposed enterprise (by generating inputs from other firms within the model itself for validation purposes). They are shown in figures 7.12, 7.13 and 7.14.

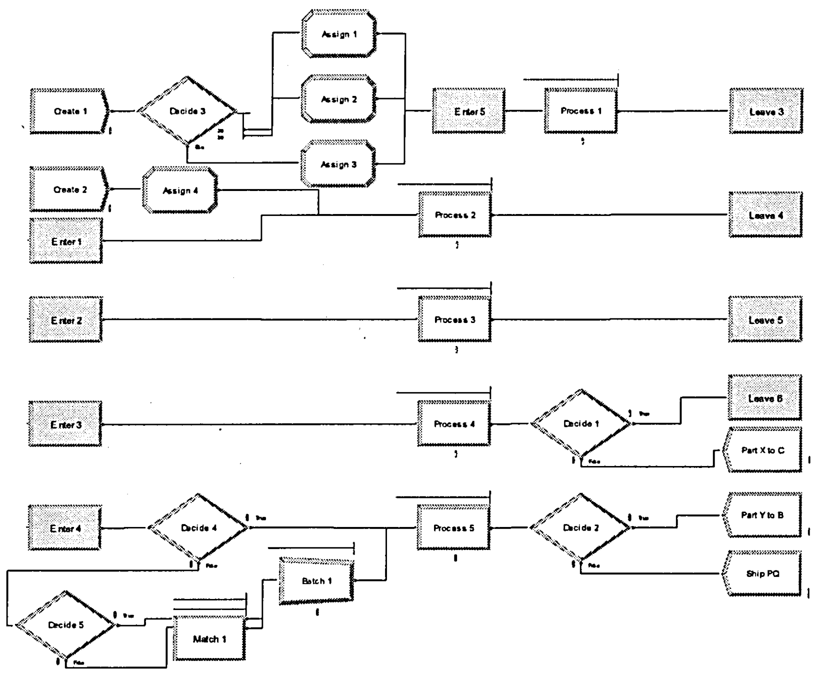


Figure 7.12 – Simulation model of Firm A

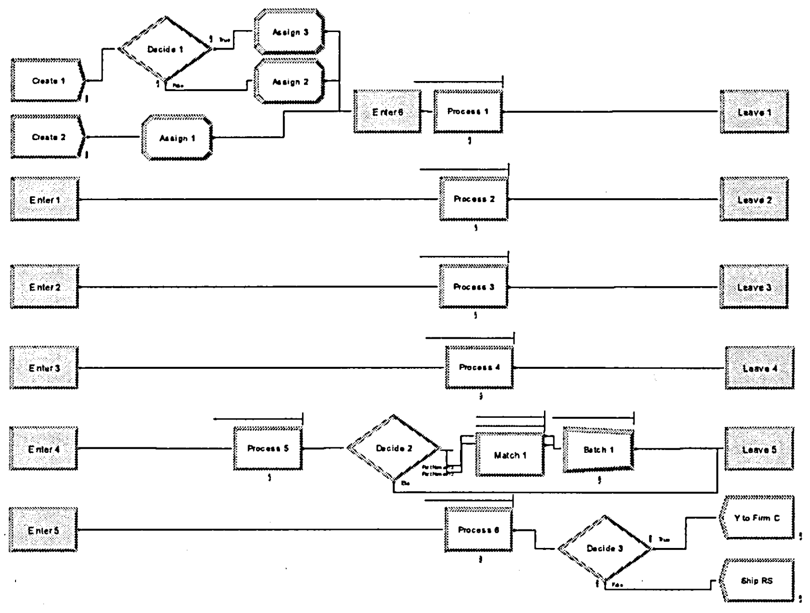


Figure 7.13 – Simulation model of Firm B

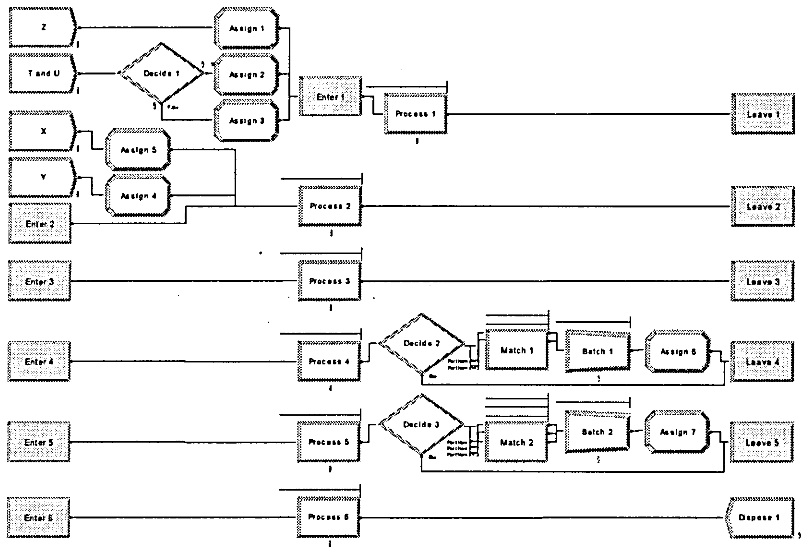


Figure 7.14 – Simulation model of Firm C

Once the 3 models were validated, they were modified to pass the output to and accept input from other models. In order to pass the output, a "VBA block" was added to the model. When an entity passes through the "VBA block", the API written for the "VBA block" (see appendix 1) extracts information from the model and sends a message to a queue of the destination model. Parameters on the parts and other information such as quantity of parts transferred as the output can be included in messages. Since batches of 100s are sent, a batch module is added just before the "VBA block". The API of the receiving model is designed in such a way that when a message arrives to a queue, an event automatically fires and extracts the information contained in the message, and passes them to the Arena simulation model. Based on the information received, entities are created and scheduled for releasing into the simulation model (as parts). In order to validate individual simulation models of firms B and C, "Create modules" were used to generate entities for representing input parts receive from other models. Once validated, for releasing parts created based on the information received from the output model, "Create modules" were replaced with "Create blocks" (with zero entities created by the block itself). This was done to facilitate the use of "EntitySendToBlockLabel" command which simplifies releasing of entities to a specific location of the simulation model. Modified simulation models of firms A, B and C are shown in figures 7.15, 7.16 and 7.27.

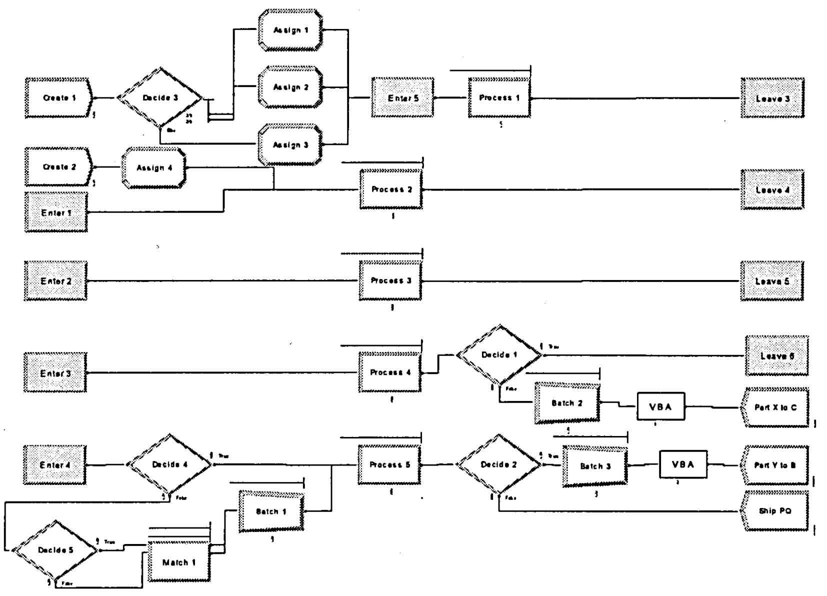


Figure 7.15 – Modified simulation model of Firm A

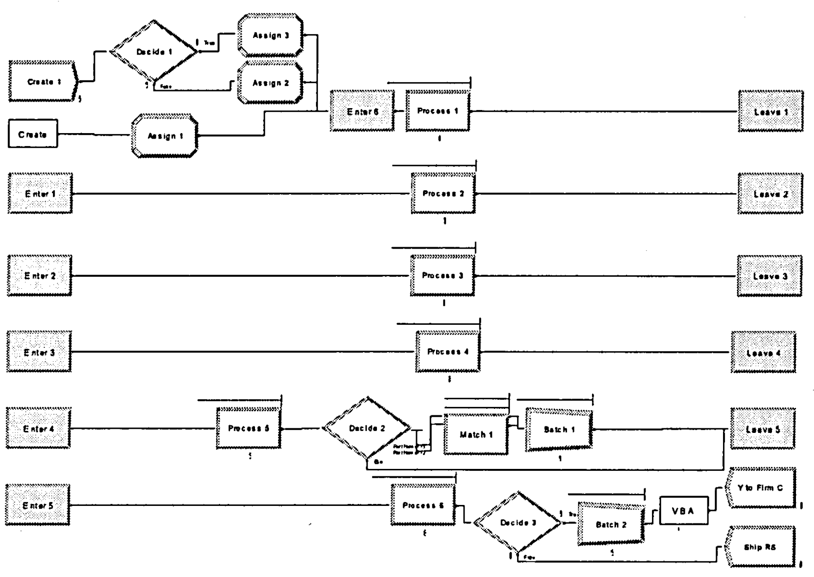


Figure 7.16 – Modified simulation model of Firm B

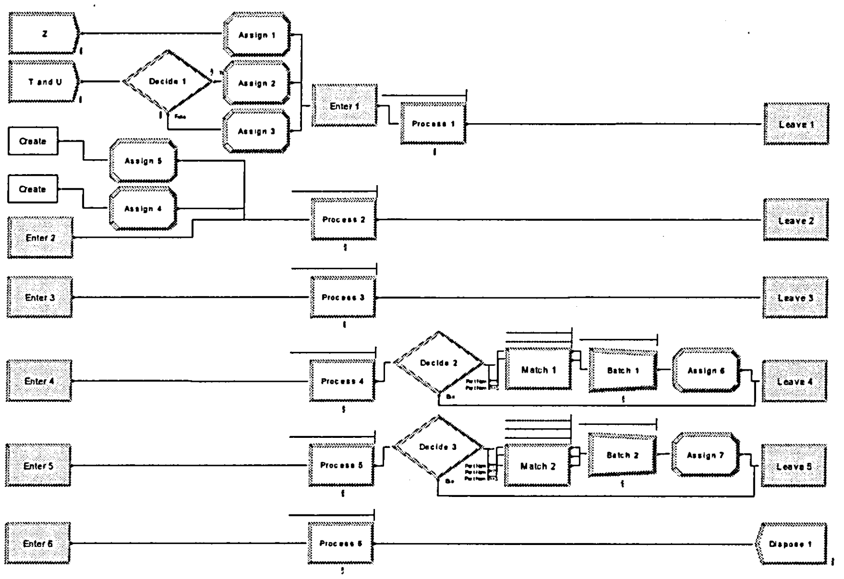


Figure 7.17 – Modified simulation model of Firm C

In order to receive messages relating to information on parts coming from other simulation models and messages relating to synchronization of the distributed simulation system, 2 queues were created in each workstation. The first queue (pq prefixed with model name) receives messages relating to work in progress information and the other queue (sq prefixed with model name) receives messages relating to the synchronization mechanism. To synchronize the distributed simulation the approximate synchronization mechanism presented in the previous chapter (Chapter 7) was incorporated into the VBA code of Arena simulation models. To receive simulation times from distributed simulation models, 3 message queues were created (tq prefixed with model name) (figure 7.18) for the time processing unit (TPU) of the synchronization mechanism. The TPU was modified to start (the loaded simulation model) and stop simulation models. It also indicates current simulation times and status of simulation models (figure 7.19).

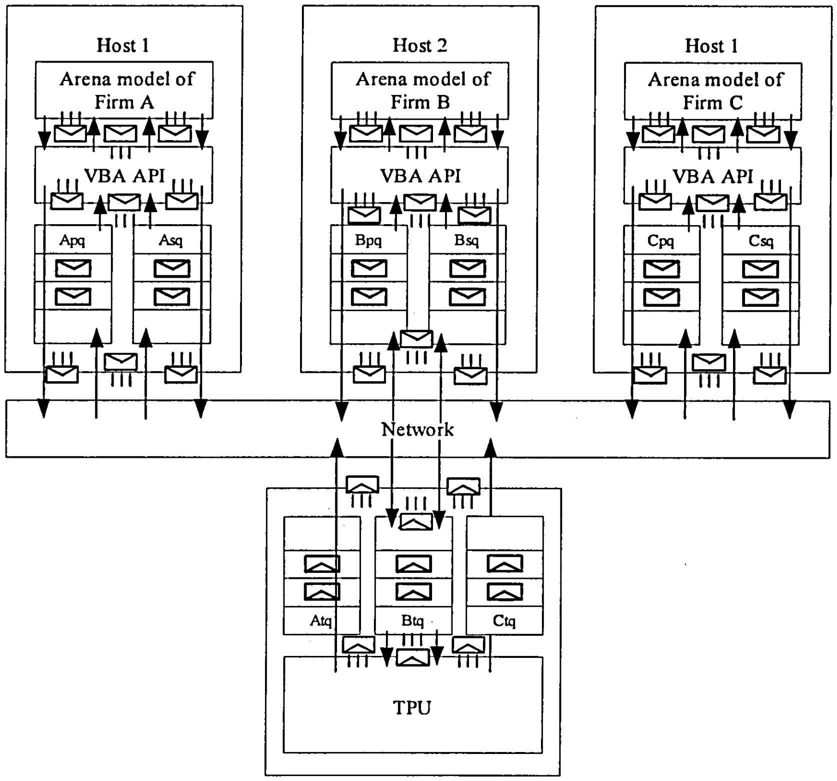
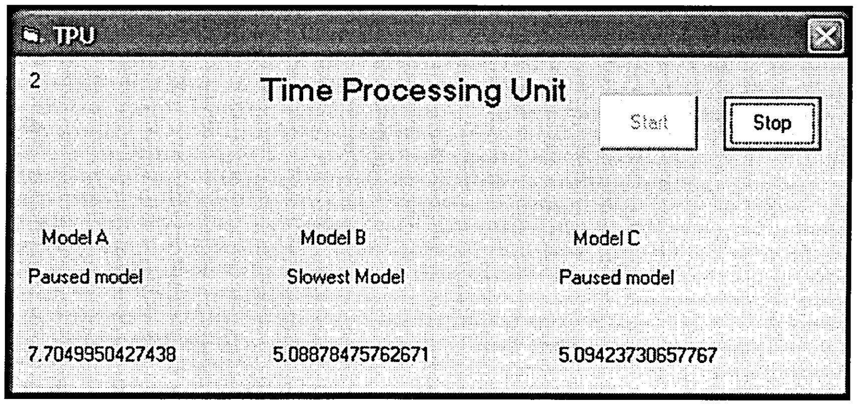


Figure 7.18 – Arena, MSMQ, API and TPU



2

Time Processing Unit

Model A
Paused model
7.7049950427438

Model B
Slowest Model
5.08878475762671

Model C
Paused model
5.09423730657767

Figure 7.19 – Modified time processing unit (TPU) of distributed simulation

7.8 Output from distributed enterprise simulation

Distributed simulations present new challenges when collecting and analyzing output from a simulation system as output is generated by distributed simulation models at different locations. With the proposed approach, the output is generated at individual simulation models distributed across the network, and for the entire system no output is generated from the distributed simulation system itself. In order to obtain output for the entire system, the distributed enterprise simulation needs to be configured (programmed) to generate the required output after identifying what parameters are needed as the output for the proposed enterprise. In addition, some of the information generated may be accessed only at local levels mainly in order to hide proprietary information. This highlights the necessity of identifying which part of the output needs to accessed locally and which needs to be integrated to reflect operations of the whole enterprise. In order to illustrate the output generation at different levels, a sample of performance analysis parameters relating to manufacturing enterprises was selected.

A wide array of performance measures including throughput, queue length, average waiting times, resource utilization, output rate, work in process (WIP), cycle time were proposed by a number of authors (Dahl and Jacob, 2000; Duwayri et al., 2001; Eneyo and Panniselvan, 1998; Law and McComas, 1999; Silva et al., 2000). In order to demonstrate the output generation process for individual models and the entire enterprise the following sample performance measures were chosen.

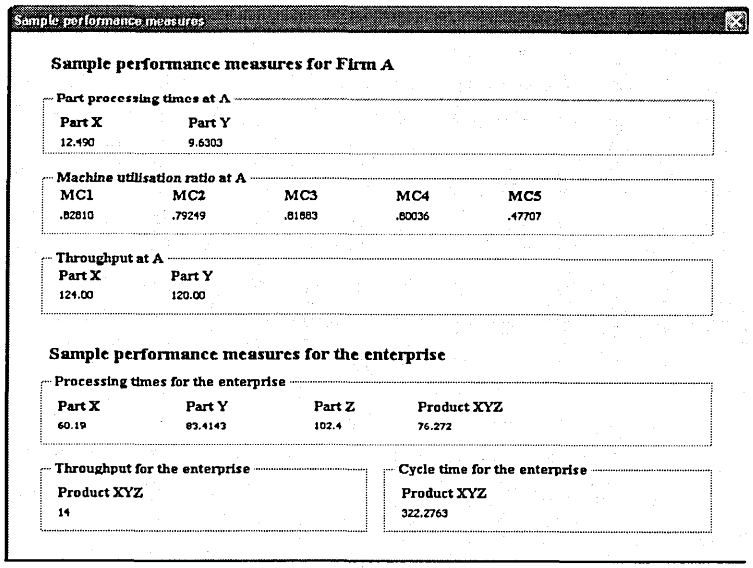
Individual models

- Cycle times for parts & components
- Machine utilization
- Throughput of parts and components

Distributed manufacturing systems

- Cycle time for products
- Throughput of the final products

Output files (<model name>.OUT) generated at individual models were used to extract relevant information to calculate parameters required (Program code used to generate output, which is integrated to API is given at Appendix 1). Performance measures relating to individual models were displayed locally together with performance measures for the entire enterprise. Figures 7.20, 7.21 and 7.22 show sample performance measures generated for three firms of the proposed enterprise.



Sample performance measures

Sample performance measures for Firm A

Part processing times at A

Part X	Part Y
12.490	9.6303

Machine utilisation ratio at A

MC1	MC2	MC3	MC4	MC5
.82810	.79249	.81883	.80036	.47707

Throughput at A

Part X	Part Y
124.00	120.00

Sample performance measures for the enterprise

Processing times for the enterprise

Part X	Part Y	Part Z	Product XYZ
60.19	83.4143	102.4	76.272

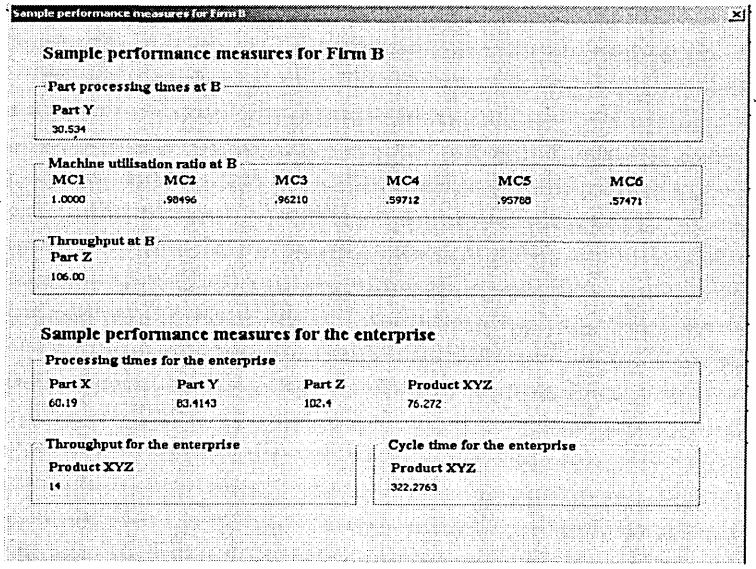
Throughput for the enterprise

Product XYZ
14

Cycle time for the enterprise

Product XYZ
322.2763

Figure 7.20 – Sample performance measures at Firm A



Sample performance measures for Firm B

Sample performance measures for Firm B

Part processing times at B

Part Y
30.534

Machine utilisation ratio at B

MC1	MC2	MC3	MC4	MC5	MC6
1.0000	.98496	.96210	.59712	.95788	.57471

Throughput at B

Part Z
106.00

Sample performance measures for the enterprise

Processing times for the enterprise

Part X	Part Y	Part Z	Product XYZ
60.19	83.4143	102.4	76.272

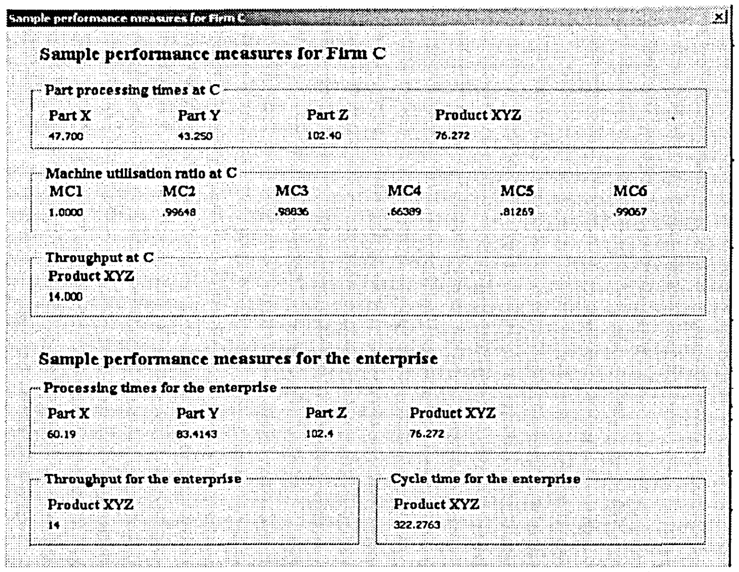
Throughput for the enterprise

Product XYZ
14

Cycle time for the enterprise

Product XYZ
322.2763

Figure 7.21 – Sample performance measures at Firm B


Sample performance measures for Firm C

Sample performance measures for Firm C

Part processing times at C

Part X	Part Y	Part Z	Product XYZ
47.700	43.250	102.40	76.272

Machine utilisation ratio at C

MC1	MC2	MC3	MC4	MC5	MC6
1.0000	.99648	.98836	.66389	.81269	.99067

Throughput at C

Product XYZ
14.000

Sample performance measures for the enterprise

Processing times for the enterprise

Part X	Part Y	Part Z	Product XYZ
60.19	83.4143	102.4	76.272

Throughput for the enterprise

Product XYZ
14

Cycle time for the enterprise

Product XYZ
322.2763

Figure 7.22 – Sample performance measures at Firm C

According to the proposed methodology for distributed enterprise simulation, the next stage is the verification and validation of the programmed distributed simulation system to make sure that the simulation system represents the system under investigation.

7.9 Validation of the distributed enterprise simulation system

Verification and validation is a well researched area in discrete event simulation. The existing techniques can be used to verify and validate the conceptual model, partitioned models, computer simulation models etc. Since existing verification and validation techniques are focused on sequential simulation, some of the mechanisms such as message passing, synchronization included in the distributed enterprise simulation required to be validated using new approaches.

7.9.1 Validation of message passing mechanism

Messages are used for passing work-in-progress between distributed simulation models and synchronize the simulation system. In order to do these, a message is created with necessary parameters at one workstation and send to the destination queue at another workstation. The validation system should make sure that the generated message reaches its destination without much delay. This process is relatively simple and straightforward as a small program can be used to indicate when a message arrives at the destination.

7.9.2 Validation of the synchronization mechanism

The approximate synchronization mechanism forces distributed simulation models to run at approximately same simulation time. The validation system should make sure that distributed simulation models run at approximately same simulation time. To validate the embedded synchronization mechanism, distributed enterprise simulation was executed with and without the synchronization mechanism at different execution speeds as shown below.

	Model A Run speed	Model B Run speed	Model C Run speed
Case 1	0.007	0.007	0.007
Case 2	0.007	0.007	0.008
Case 3	0.007	0.008	0.007
Case 4	0.008	0.007	0.007

Table 7.4 – Run speeds to validate the working of the distributed enterprise simulation with the approximate synchronization mechanism

Case 1

All three models were executed at the same run speed (0.007). Figures 7.23 and 7.24 show distributed enterprise simulation without and with the approximated synchronization mechanism respectively.

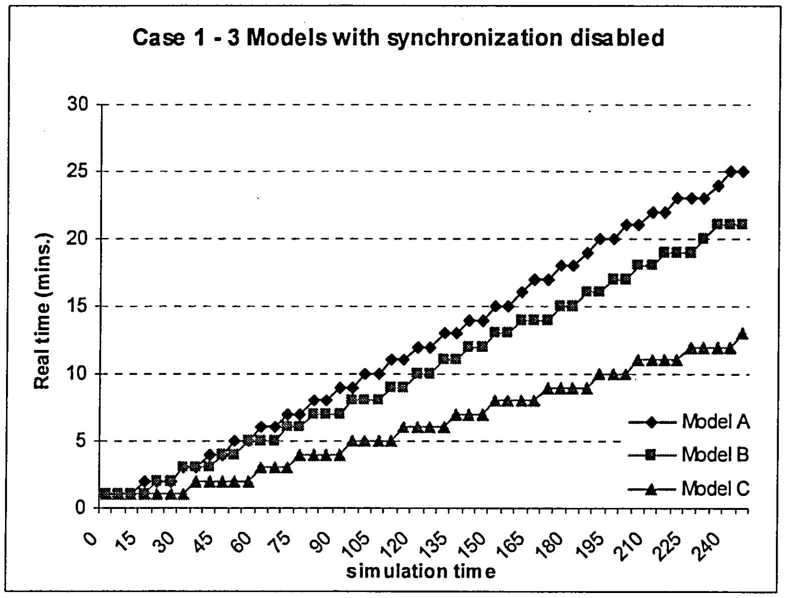


Figure 7.23 – Three models without the approximate synchronization mechanism

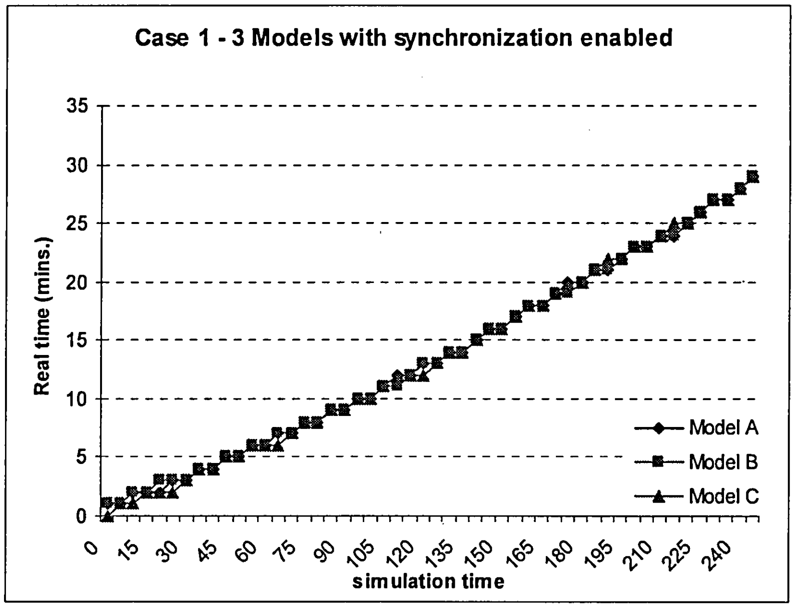


Figure 7.24 – Three models with the approximate synchronization mechanism

Case 2

Models A B and C were executed at 0.007, 0.007 and 0.008 run speeds respectively. Figure 7.25 shows distributed enterprise simulation without the approximated synchronization mechanism and figure 7.26 shows the same simulation with the synchronization mechanism enabled.

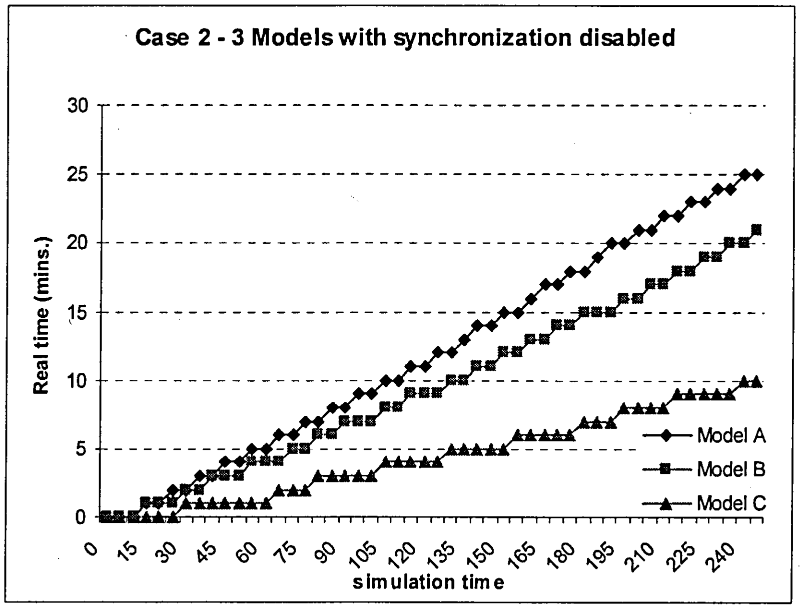


Figure 7.25 – Three models without the approximate synchronization mechanism

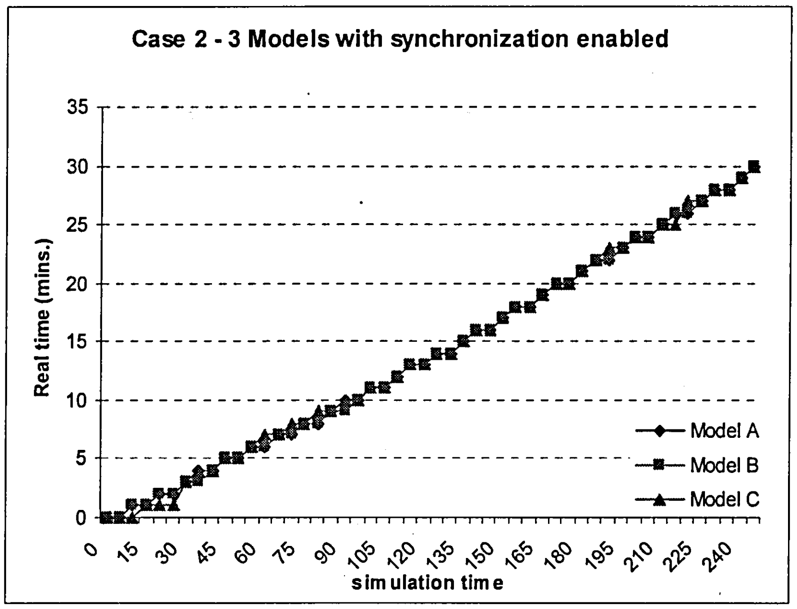


Figure 7.26 – Three models with the approximate synchronization mechanism

Case 3

Models A B and C were executed at 0.007, 0.008 and 0.008 run speeds respectively. Figure 7.27 shows distributed enterprise simulation without the approximated synchronization mechanism and figure 7.28 shows the same simulation with the synchronization mechanism enabled.

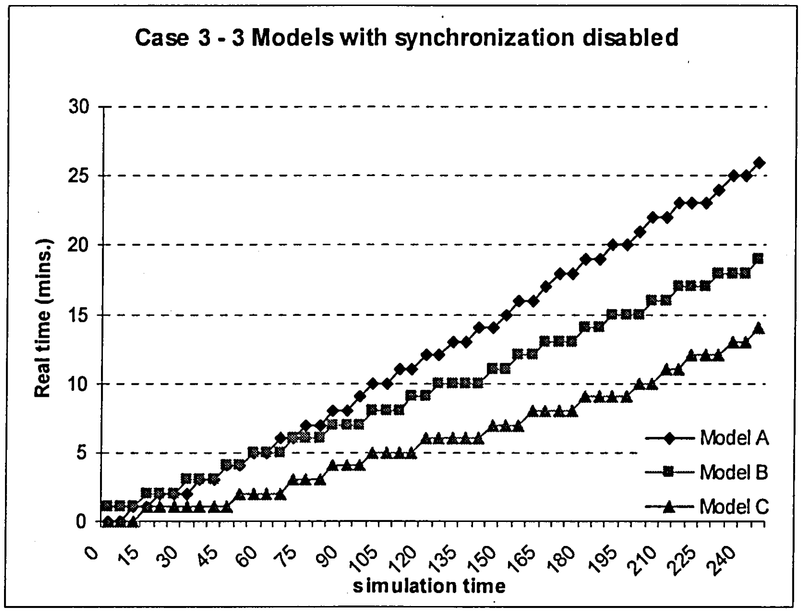


Figure 7.27 – Three models without the approximate synchronization mechanism

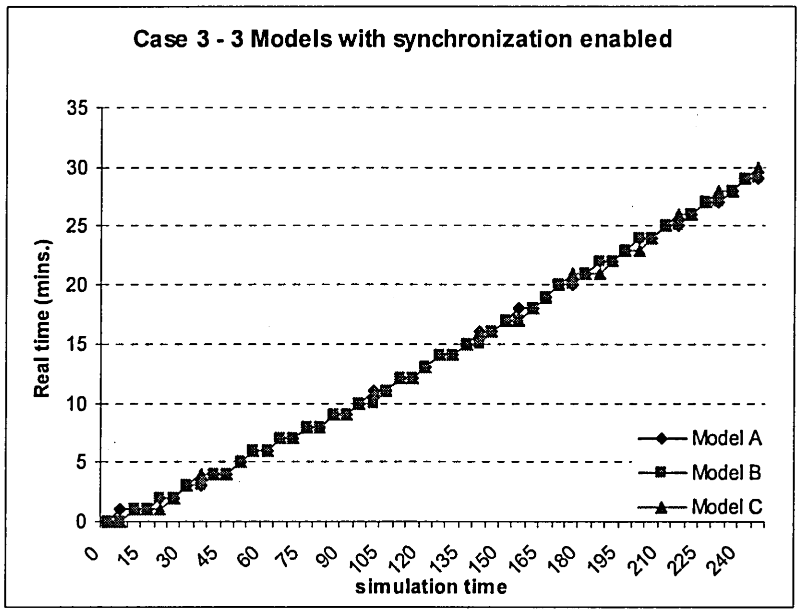


Figure 7.28 – Three models with the approximate synchronization mechanism

Case 4

Models A B and C were executed at 0.008, 0.007 and 0.007 run speeds respectively. Figure 7.29 shows distributed enterprise simulation without the approximated synchronization mechanism and figure 7.30 shows the same simulation with the synchronization mechanism enabled.

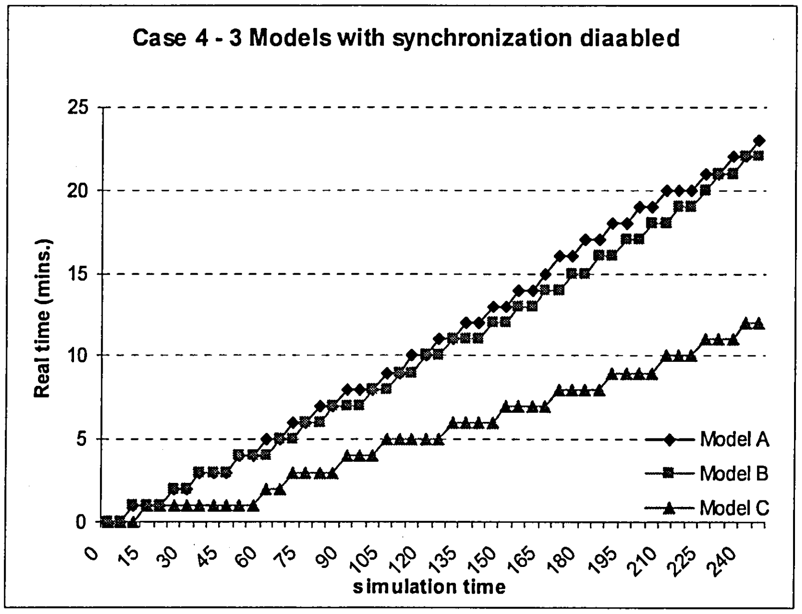


Figure 7.29 – Three models without the approximate synchronization mechanism

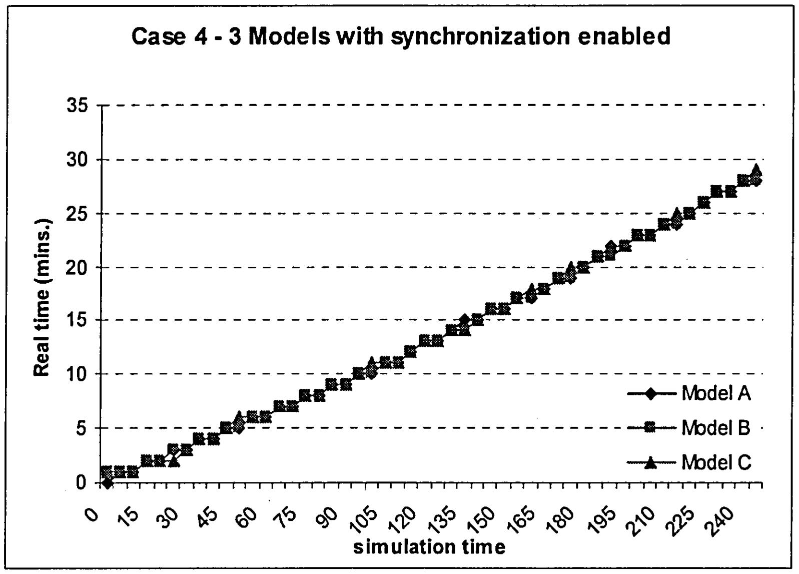


Figure 7.30 – Three models with the approximate synchronization mechanism

Accordingly, it shows that for any run speed the approximate synchronization mechanism forces distributed simulation models to run approximately at the same simulation time. This validates the approximate synchronization mechanism used for distributed enterprise simulation.

Summary

This chapter presented detailed work relating to the implementation process of the distributed enterprise simulation. As it was noted in Chapter 1, commercial simulation software, and simple and cost effective technologies were utilized to implement the distributed simulation. This was done in order to simplify the implementation processes and to address some of the criticisms directed towards distributed simulation due to its complexity and high cost to implement, need for more expertise etc. A hypothetical case study was used to illustrate the implementation. Arena simulation software, VBA and MSMQ were used to implement the hypothetical case study presented. Experimentation (warm-up period, number of replications, speed of the simulation etc.), analysis of results generated from the simulation, and decisions taken based on output analysis can be implemented once it was established that the distributed enterprise simulation system is valid.

Chapter 8

Discussion, conclusions and suggestions for further work

This is the concluding chapter of the thesis. Issues raised in previous chapters are discussed in this section. In addition, it also presents conclusions reached from the research and, provides suggestions to enhance the methodology for distributed enterprise simulation and to continue the research further.

8. 1 Introduction

The thesis presented a novel approach for developing enterprise simulation models using distributed simulation. It includes a methodology for distributed enterprise simulation on which the proposed approach centres and detailed approaches:

- For selecting an appropriate simulation strategy
- For conceptual modelling, model partitioning and mapping
- To synchronize a distributed enterprise simulation system
- For implementing the distributed enterprise simulation system

Although research in parallel and distributed simulation has been carried out for more than two decades, analysis of the literature shows that still the general simulation community failed to appreciate it fully. The complexity, time and cost involved in developing, need for more expertise, lack of a proper methodology available for developing parallel and distributed simulation etc. are few of the reasons highlighted in the literature for this lack of acceptability. Some of these points acted as motivating factors when developing the proposed approach for distributed enterprise simulation.

In addition to the widely investigated areas of (parallel and) distributed simulation such as synchronization, this research also explored message passing mechanisms and simulation software. Contributions from the thesis for the field of parallel and distributed simulation include the above mentioned methodology for distributed enterprise simulation, the simulation strategy selection (SimSS) process, a simplified approach for model partitioning and mapping, the approximate synchronization mechanism, and a simplified approach to implement distributed enterprise simulations. The following sections provide brief discussions of these. Furthermore, it is expected that the proposed simplified approaches address some of the criticisms directed towards distributed simulation due to its failure to penetrate into general simulation applications. The next section offers a discussion of key stages of the proposed methodology which was presented in previous chapters. Section 8.3 provides conclusions reached from the researched carried out. Suggestions for further work in order to improve the research already carried out are presented in section 8.4.

8.2 Discussion

8.2.1 The proposed methodology for distributed enterprise simulation

Analysis of the literature suggested that unlike sequential simulations, the field parallel and distributed simulation lacks formal methodologies for developing such simulations. A number of authors highlighted the need for a formal methodology. It is expected that the proposed methodology may fulfil this need.

The proposed methodology for distributed enterprise simulation was derived by combining additional activities required for parallel and distributed simulation with activities required for traditional sequential simulation. Although it was developed focusing on distributed enterprise simulation, generally it can be also applied when developing distributed simulations and up to some extent parallel simulations as well. However, after model partitioning and mapping stage, parallel simulation requires different approaches in mapping, programming, message passing etc. The proposed methodology is not a purely sequential process, some preceding stages need revising if it is found at a particular stage that the proposed model/system does not reflect the system under investigation, project objectives are not going to be met or system under investigation changed. It is not expected that simulationists will strictly adhere to the proposed methodology, but will use it as a set of guidelines when developing distributed enterprise simulations.

The main benefit of the proposed methodology is that it provides a set of predefined stages to follow when developing distributed enterprise simulations thus reducing the associated complexities and simplifying the development process. The methodology is especially useful as distributed simulations are inherently more complex than sequential simulations. Validity of the distributed simulation system can be enhanced as verification and validation carried out at three stages of the proposed methodology. Moreover, this also simplifies the verification and validation process too. However, applicability of the proposed methodology (particularly latter stages) to develop highly complex systems such as logic gates, telecommunication systems, computer networks etc. needs further investigation.

8.2.2 The simulation strategy selection (SimSS) process

The proposed simulation strategy selection (SimSS) process helps users to determine an appropriate simulation strategy out of sequential simulation, parallel simulation and distributed simulation. The main factor that motivated to present this process is the complexity of (parallel and) distributed simulation. Although parallel and distributed simulation provides an attractive alternative for conventional sequential simulation when simulating large and complex systems, the former is more complicated, effort intensive, costly and requires more expertise. Simulation model developers and users need to carefully consider costs and benefits of using parallel or distribution simulation before making decision to use it.

The analytic hierarchical process (AHP) provides the basis for the SimSS process. The main reasons to use the AHP include: its ability to incorporate subjective criteria into the decision making process as well as simplicity and availability of the AHP based software. Expert Choice which provides a simple and user friendly interface was chosen for calculating and ranking alternatives. However, calculations and ranking can be done manually without using Expert Choice too.

The SimSS process presents three alternatives namely: sequential simulation, parallel simulation, and distributed simulation; and four criterions for evaluating alternatives namely: execution time, computational resources, complicated model development process, and need to execute in geographically distributed manner. These criterions were widely cited in the literature as factors that motivate users for employing parallel or distributed simulation. First three factors can be considered as encouraging factors while the last one acts as a deciding factor. If a simulation model for the system under investigation requires more computational resources or too complicated to develop as a single model, either parallel or distributed simulation can be used. However, if simulation needs to be executed in geographically distributed environment then distributed simulation is the only solution available. Users are required evaluating alternatives using a criteria based on requirements, resources available and situation. Although the SimSS process does not provide a definite solution, it prioritises alternatives based on user's evaluations. It was illustrated in chapter 5 that the simulation strategy depends on situational factors such as availability computational resources, required execution speed for the simulation, users' and modellers' preferences, availability of expertise etc.

The main advantage of the SimSS process is that it guides simulationists through the decision making process in order to select an appropriate simulation strategy. It also prevents users employing parallel or distributed simulation unnecessarily thus saving time, cost and resources. However, the process can be further improved by incorporating additional factors such as availability of expertise, resources etc into the criterion.

8.2.3 Model representation, partitioning and mapping

Model partitioning and mapping are two additional activities which need to be carried out for (parallel and) distributed simulation when compared with sequential simulation. Factors that affect performance and efficiency of distributed simulation such as size of a logical process, balance of load among processors, number of messages pass among processors etc. are depend on how the entire simulation model is partitioned and mapped. Furthermore, a distributed simulation may also affect performance of the computer network on which it runs with network traffic generated by the simulation. Therefore, mapping and partitioning is one of the stages that require careful attention when implementing a distributed simulation as it affects performance of both distributed simulation itself and the computer network. However, unlike areas such as synchronization, the literature has not paid much attention to this important area. Furthermore some of the algorithms presented in the literature require developing the distributed simulation programme, executing it sequentially and collecting data for the purpose of partitioning and mapping the simulation model. Although this process may help to implement an efficient and high performing distributed simulation, it also leaves simulation users in a dilemma due to the fact that distributed simulation is used only because the simulation can not be run as a sequential simulation.

The new approach for model representation, partitioning and mapping was presented to address some of the issues noted above. The main difference between the proposed approach and existing approaches is that the new approach proposes to partition the conceptual model (of the system under investigation) into logical processes, then develop simulation models for these partitioned logical processes. It proposes to use IDEF0 technique for model representation as it is one of the widely used model representation technique in simulation. Furthermore, sub sections that can function independently can easily be identified in the IDEF0 model. At detailed abstraction level, developing the simulation model from the IDEF0 model is relatively straightforward as processes/ functions of the model can be represented by blocks and modules of Arena simulation software which was used to develop distributed simulation models. If the conceptual model is developed using Microsoft Visio, the IDEF0 model can be directly converted into an Arena simulation model.

With the proposed approach, mapping is relatively straightforward and simple since only one logical process is assigned to one networked workstation. However, with this approach efficiency of the distributed enterprise simulation may affect due to load balancing problems.

The ability to develop distributed simulation models without first developing and executing the entire model as a single simulation model, and simplification of model partitioning and mapping process are the main benefits of the proposed approach. However, this approach may not appropriate when developing distributed simulations for highly complex systems as it was developed focusing on distributed enterprise simulation and especially on distributed manufacturing enterprises.

8.2.4 The approximate synchronisation mechanism

As noted in the chapter 5, synchronisation is one of the well researched areas in parallel and distributed simulation. Also synchronisation is one of the factors that makes distributed simulation (along with parallel simulation) more complicated. Traditionally, synchronisation mechanism is integrated into the simulation program itself which enables it to control the behaviour of the distributed simulation system, especially in optimistic synchronisation protocol which requires state saving and rolling back to previous simulation times. However with the proposed approach, it was difficult to integrate the synchronisation mechanism into the core simulation program as commercial simulation software (which doesn't allow changing the simulation engine) was used to implement the distributed simulation. In the proposed approach, a part of the program code that used to synchronize simulation models was included with the application program interface (API). API is also responsible for passing messages between distributed simulation models. On the other hand, also it can be argued that the API is part of the simulation model as it was programmed using Visual Basic for Applications (VBA) which is integrated into Arena simulation software.

Synchronization mechanisms make sure that messages from distributed simulation models are executed in the timestamped order and not in the order of arrival. However, some applications of distributed simulation such as distributed manufacturing do not required to be executed in strictly synchronized environment or not needed to be synchronized at all. The proposed approximate synchronisation mechanism is appropriate for systems which do not need a strictly synchronised environment. It was developed based on conservative synchronization protocol as it is difficult to implement a state saving and roll back mechanism (with commercial simulation software) which is the basis for the optimistic synchronization approach. As name implies the approximate synchronization mechanism does not enforce strictly synchronised environment. Instead, it forces distributed simulation models to run approximately at the same simulation time.

The proposed mechanism is less complicated to implement than conventional synchronization mechanisms. The main benefit of the approximate synchronization approach is its simplicity to develop and implement. Moreover, additional distributed simulation models can be incorporated into the system with slight modifications as modules that use for the approximate synchronisation mechanism are independent of the main simulation model. As previously presented approaches relating to simulation strategy selection and, model representation, partitioning and mapping; it is expected that the proposed approximate synchronization mechanism also addresses the criticism made towards (parallel and) distributed simulation due to its complexity for implementing. However, it should be noted that the proposed approach may not suitable for all circumstances, especially if simulation requires running in strictly synchronized environment. In addition, the mechanism may also less efficient when compared to other approaches presented in the literature.

8.2.5 Implementation approach

As noted in previous sections distributed simulation is more complicated to implement than sequential simulation, involving long development times, higher costs, steep learning curves and requiring more expertise. Mainly due to these reasons this type of simulation is not much used in general industrial and business applications. The proposed approached employed technologies and software that make the implementation process of distributed enterprise simulation relatively less complicated and cost effective.

Using of programming languages such as Java, C++ or Smalltalk may complicate the implementation process of the DMS. This calls for expertise not only in distributed simulation but also in computer programming resulting higher costs and longer development times. Moreover message passing middleware has to be procured and customised to the distributed simulation system incurring additional costs. The proposed approach employs commercial simulation software for developing simulation models. Although these packages too are expensive, it can be expected that most of the organisations which intend to use distributed simulation already use commercial simulation software packages to simulate their operations. Use of commercial simulation software also simplifies the simulation model development process resulting minimum additional expenses. Analysis of the recent literature too shows attempts to use commercial simulation software in distributed simulation especially in distributed supply chain simulation. Microsoft message queue (MSMQ) also offers a cost effective solution as a message passing middleware. Since MSMQ is integrated into Windows 2000 (both professional and server) and Windows XP operating systems, and also supports Windows 98 and 95 as a free add-on saves the cost of middleware. Visual basic for applications (VBA) was used as the application program interface (API) that interacts between

MSMQ and Arena which was chosen as the commercial simulation software for illustrating the implementation process. However, other commercial simulation packages such as Automod, Promodel, Witness etc. too can be used instead of Arena. Most of these packages support either or both C++ or/and VBA. As VBA is integrated into both MSMQ and Arena, it was decided to use VBA instead of C++. VBA offers a similar programming interface as popular Visual Basic programming language and, also easy to learn and requires less expertise in programming.

Simplified and cost effective implementation of distributed enterprise simulation is the main benefit of the proposed approach. However, it also encourages reusability of existing simulation models developed with commercial simulation software. With slight modifications, these models can be adapted into a distributed simulation thus saving model development time and cost. Since distributed simulation models interact only through messages and functionality of one model is independent of another, it is possible to integrate simulation models developed with different simulation software and/ or with API developed with different programming language such as C++, as long as MSMQ uses for message passing. This provides an opportunity for different enterprises which use different simulation software packages for implementing distributed enterprise simulation with existing models and/ or new models without purchasing another package in order to use the same simulation software.

Animation is highly useful to visualise the working (ie.. system under investigation) and results of the simulation for managers and employees of an organisation. In addition, the ability to see the simulation activities while a simulation is running offers several more advantages. Users can observe trends that cannot be captured using average statistics (that are typically available only at the end of the simulation run). Furthermore, visualization allows user to take immediate corrective measures on the model, instead of waiting until the simulation ends, if a modelling problem is observed. This is particularly crucial for a distributed enterprise simulation since simulation of this scale takes relatively longer time to complete. Animation capabilities of purpose built distributed simulators developed with C++ or java may not as effective as animation capabilities provided by commercial simulation software packages. This is another advantage of using commercial simulation software.

However, animation effects slow down distributed simulation so that speedups expected to gain through distributed simulation may not be able to achieve with the proposed implementation approach. Although speedup of simulation is one of the main reasons to use distributed simulation, it is not the only factor to use such simulations. In addition to points noted in previous paragraphs, ability to hide confidential or proprietary information, provide more computational resources, and to obtain simulation results distributed manner are some of the reasons that encourage the use of distributed simulation. However, the proposed implementation approach may not appropriate to implement highly complex systems such as logic circuits, telecommunication systems, computer networks etc.

8.3 Conclusions

Based on the discussion presented above, following conclusions can be reached:

- The proposed methodology for distributed enterprise simulation streamlines and simplifies the development process of enterprise simulations.
- The approximate synchronization mechanism presents a less complicated synchronisation approach for some distributed simulation applications.
- Use of commercial simulation software, Microsoft Message Queue (MSMQ) and Visual Basic for Applications (VBA) made the implementation process simple and cost effective.
- It is also expected this simplified and cost effective approach may address some of the criticisms made towards distributed simulation due to its complexity and association with high costs.
- Due to animation effects and the approximate synchronisation mechanism, speedups expected to gain with distributed simulation may not be able to achieve.
- It may not be possible to use the proposed approach to develop distributed simulations for highly complex and large systems.

8.4 Suggestions for further work

The proposed methodology for distributed enterprise simulation was developed with a view of simplifying the simulation model development process and also cutting time and cost involved. However, due to restrictions of time, some of the additions and refinement to the proposed approach were not able to accomplished. In order to improve the methodology and also to understand advantages and disadvantages of it further, followings are suggested:

As noted in the section 8.2.3, if Microsoft Visio is used to develop IDEF0 model then it can be directly converted into Arena simulation model. This may simplify the model development process, as it shortens the distributed enterprise simulation development time and calls for less expertise. Therefore, it is suggested to implement a system to directly convert the IDEF0 model into an Arena simulation model.

The proposed approach utilised MSMQ and a commercial simulation software package to develop the distributed enterprise simulation. Most of distributed simulations were developed using programming languages such as C++, Java, smalltalk etc and using middleware such as CORBA and HLA. It is desirable to compare performance of distributed simulations developed using the proposed approach and conventional approaches. This may help to determine in which situations the proposed approach can be employed to implement distributed enterprise simulations. It is also suggested to compare performance of different message passing middleware with simulation models developed using commercial simulation packages.

For the approximate synchronization mechanism it is worth investigating the effect of changing the time interval between requesting of simulation times (from distributed simulation models) by time processing unit (TPU) on occurrence of synchronization errors. Furthermore, it is also desirable to examine how a distributed enterprise simulation developed using commercial simulation software, MSMQ and VBA operates in a strictly synchronised environment.

One of the most important aspects of any simulation is to collect the output to determine and compare performance parameters. Distributed simulation presents new challenge on collecting and analysing the output as output generated in distributed manner. Chapter 7 briefly looked into how the output is generated at different simulation models are collected and integrated to present performance measures for the whole system. However, it is desirable to develop a comprehensive mechanism to present performance measures and other output measures once the distributed simulation is completed and also while it is running if necessary.

This system can be further improved by developing additional mechanism to change some parameters that determine working of the model such as process times, delay times, schedules and sequences etc. from a central location rather than editing simulation models individually. This may be especially useful if models are located at different physical locations. However, this may not be possible if distributed simulation was selected in order to hide any proprietary information.

For the purpose of message passing MSMQ 2.0 was used for the research. However MSMQ 3.0 allows message passing through HTTP protocol. This provides exciting new opportunities to run distributed enterprise simulation models which are connected each other through the internet. As web based simulation is a emerging area in simulation, it is desirable to investigate how distributed enterprise simulation systems developed with commercial simulation software executed over the internet using message passing tools such as MSMQ.

Summary

The concluding chapter of this thesis presented a discussion of research carried out and conclusions reached. In addition, it also offered suggestions in order to improve the research work already conducted.

References

Abeysunadara, B. W. and A. E. Kamal 1991. High speed local area networks and their performance: A survey, *ACM Computing surveys*, vol. 23, no. 2, 221-264.

Abrams, M. 1993. Parallel discrete event simulation: fact or fiction, *ORSA Journal on Computing*, vol. 5, no. 3, 231-233.

Al-Ahmari, A. M. A. and K. Ridgeway 1999. An integrated modelling method to support manufacturing systems analysis and design, *Computers in Industry*, vol. 38, no. 3, 209-223.

Al-Habri, K. M. A.-S. 2001. Application of the AHP in project management, *International Journal of Project Management*, vol. 19, no. 1, 19-27.

Alfieri, A. and P. Brandimarte 1997. Object-oriented modelling and simulation of integrated production/ distribution systems, *Computer Integrated Manufacturing Systems*, vol. 10, no. 4, 261-266.

Antony, J. 1998. Some key things industrial engineers should know about experimental design, *Logistics Information Management*, vol. 11, no. 6, 386-392.

Bagrodia, R. L. 1996. Perils and pitfalls of parallel discrete-event simulation. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 136-143. Coronado, CA, USA.

Bakken, D. E. 2003. Middleware, in *To appear in Encyclopaedia of Distributed Computing*, ed. J. Urban and P. Dasgupta, Kluwer Academic Press.

Balci, O. 1990. Guidelines for successful simulation studies. In *Proceedings of 1990 Winter Simulation Conference*, ed. O. Balci, R. P. Sadowski, and R. E. Nance. 25-32. New Orleans, LA, USA.

Balci, O. 1998. Verification, validation and accreditation. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 41-48. Washington DC, USA.

Baldwin, L. P., T. Eldabi, V. Hlupic, and Z. Irani 2000. Enhancing simulation software for use in manufacturing, *Logistics Information Management*, vol. 13, no. 5, 263-270.

Ball, P. 1998. Abstracting performance in hierarchical manufacturing simulation, *Journal of Materials Processing Technology*, vol. 76, 246-251.

Banks, J., Carson, J. S., and B. L. Nelson 1996. *Discrete Event System Simulation* Prentice-Hall.

Banks, J. 2000. Introduction to simulation. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 9-16. Orlando, FL, USA.

Banks, J., Carson, J. S., Nelson, B. L., and D. M. Nicol 2000. *Discrete Event System Simulation* Prentice-Hall, New Jersey.

Barton, R. R. 2001. Designing simulation experiments. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 47-52. Arlington, VA, USA.

Bass, J. 1999. Design environments for parallel and distributed processing. In *Proceedings of 1999 Euromicro Workshop on Parallel and Distributed processing*, Funchal, Portugal.

Benjamin, P. C., M. Erraguntla, D. Delen, and R. J. Mayer 1998. Simulation modeling at multiple levels of abstraction. In *Proceedings of 1998 Winter Simulation Conference*, ed. D. J. Medeiros, E. F. Watson, J. S. Carson, and M. S. Manivannan. 391-398. Washington DC, USA.

Berson, A. 1996. *Client Sever Architecture* McGraw-Hill.

Borah, J. 2000. Conceptual modelling - How do we move forward? - The next step. In *Proceedings of 2002 Simulation Interoperability Workshops,*

Borah, J. 2002. Conceptual modelling - The missing link of simulation development. In *Proceedings of 2002 Simulation Interoperability Workshops,*

Boukerche, A. and C. Trooper 1994. A static partitioning and mapping algorithm for conservative parallel simulations. In *Proceedings of 1994 Workshop on Parallel and Distributed Simulation.* 164-172. Edinburgh, Scotland, United Kingdom.

Boukerche, A. and A. Fabbri 2000. Reducing rollbacks through partitioning in PCS parallel simulation, *Simulation*, vol. 75, no. 1, 43-55.

Boukerche, A. and C. Trooper 2001. Local versus global lookahead in conservative parallel simulations, *Parallel Computing*, vol. 27, no. 8, 1033-1055.

Brandimarte, P. and M. Cantamessa 1995. Methodologies for designing CIM systems: A critique, *Computers in Industry*, vol. 25, no. 3, 281-293.

Bray, M., 2003. Middleware, http://www.sei.cmu.edu/str/descriptions/-middleware. html, Last accessed in June 2003.

Buss, A. 1996. Modelling with event graphs. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 153-160. Coronado, CA, USA.

Buss, A. and L. Jackson 1998. Distributed simulation modeling: A comparison of HLA, CORBA, and RMI. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 23-29. Washington DC, USA.

Cabillic, G. and I. Puaut 1997. Stardust: An environment for parallel programming on networks of heterogeneous workstations, *Journal of Parallel and Distributed Computing*, vol. 40, no. 1, 65-80.

Cagno, E., F. Caron, and A. Perego 2001. Multi-criteria assessment of the probability of winning in the competitive bidding process, *International Journal of Project Management*, vol. 19, no. 6, 313-324.

Cai, W. and Y. M. Teo 1999. Parallel and distributed simulation minitrack. In *Proceedings of 1999 Hawai International Conference on System Science*.

Calinescu, R. 1995. *Conservative discrete event simulation on bulk synchronous parallel architecture*, Programming Research Group, Oxford University Computing Laboratory, PRG-TR-16-95.

Calinescu, R. 1996, *Bulk synchronous parallel algorithms for optimistic discrete event simulation*, Programming Research Group, Oxford University Computing Laboratory, PRG-TR-8-96.

Carothers, C. D., B. Topol, R. M. Fujimoto, J. T. Stasko, and V. Sunderam 1997. Visualising parallel simulations in network computing environments: A case study. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 110-117. Atlanta, GA, USA.

Carothers, C. D. 1999. Joint special issue on parallel and distributed simulation, *Transactions of the Society for Computer Simulation International*, vol. 16, no. 1, 1-2.

Carothers, C. D., B. Topol, R. M. Fujimoto, J. T. Stasko, and V. Sunderam 1999. Visualising parallel simulations that execute in network computing environments, Future Generation Computer Systems, vol. 15. no. 4, 513-529

Carson, J. S. 1992. Modelling. In *Proceedings of 1992 Winter Simulation Conference*, ed. R. C. Crain, J. R. Wilson, J. J. Swain, and D. Glodsman. 82-87. Arlington, VA, USA.

Carson, J. S. 2002. Model verification and validation. In *Proceedings of 2002 Winter Simulation Conference*, ed. J. L. Snowdon, J. M. Charnes, E. Yucesan, and C. Chen. 52-58. San Diego, CA , USA.

Cassel, R. A. and M. Pidd 2001. Distributed discrete event simulation using the three-phase approach and Java, *Simulation: Practice and Theory*, vol. 8, no. 8, 491-507.

Centeno, M. A. and M. F. Reyes 1998. So you have your model: What to do next? A tutorial on simulation output analysis. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 23-29. Washington DC, USA.

Ceric, V. 1994. Hierarchical abilities of diagrammatic representation of discrete event simulation models. In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 589-594. Orlando, FL, USA.

Chan, F. T. S., R. W. L. Ip, and H. Lau 2001. Integration of expert systems with analytic hierarchy process for the design of material handling equipment selection system, *Journal of Materials Processing Technology*, vol. 116, no. 2-3, 137-145.

Chandy, K. M. and J. Misra 1981. Asynchronous distributed simulation via a sequence of parallel computers, *Communications of the ACM*, vol. 24, no. 4, 198-206.

Chappell, D. 1998. Microsoft message queue is a fast, efficient choice for your distributed applications, *Microsoft Systems Journal,* July, 1998

Cheng-Leong, A. 1999. Enactment of IDEF models, *International Journal of Production Research*, vol. 37, no. 15, 3383-3397.

Cheng-Leong, A., K. L. Pheng, and G. R. K. Leng 1999. IDEF*: a comprehensive modelling methodology for the development of manufacturing enterprise systems, *International Journal of Production Research*, vol. 37, no. 17, 3839-3859.

Chiola, G. and A. Ferscha 1993. Distributed simulation of Petri nets, *IEEE Parallel and Distributed Technology*, vol. 1, no. 3, 33-50.

Chow, A. H. and B. P. Zeigler 1994. Parallel DEVS: A parallel, hierachical, modular modeling formalism, In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 716-722. Orlando, FL, USA.

Cloutier, J., E. Cerny, and F. Guertin 1997. Model partitioning and the performance of distributed timewarp simulation of logic circuits, *Simulation: Practice and Theory*, vol. 5, no. 1, 83-99.

D'Souza, A. and S. K. Khator 1994. A survey of Petri net applications in modelling controls for automated manufacturing systems, *Computers in Industry*, vol. 24, no. 5, 5-16.

Dado, B., P. Menhart, and J. Safarik 1993. Distributed simulation: A simulation system for discrete event systems, *IFIP Transactions (Computer Science and Technology)* 343-353.

Dahl, T. A. and B. F. Jacob 2000. Confident decision making and improved throughput for cereal manufacturing with simulation. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 1329-1332. Orlando, FL, USA.

Das, S. R. 1996. Adaptive protocols for parallel discrete event simulation. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 186-193. Coronado, CA, USA.

Das, S. R. 2000. Adaptive protocols for parallel discrete event simulation, *Journal of Operational Research Society*, vol. 51, 385-394.

Datar, M. M. 2000. Enterprise simulation: Framework for strategic applications. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 2010-2014. Orlando, FL, USA.

Davis, W. J. 1999. Simulation: Technologies in the new millennium. In *Proceedings of Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard. 141-147. Phoenix, AZ, USA.

Dewire, D. T. 1997. *Second Generation Client/ Server Computing* McGraw-Hill.

Doumeihgts, G., B. Vallespir, and D. Chen 1995. Methodologies for designing CIM systems: A survey, *Computers in Industry* no. 25, 263-280.

Duwayri, Z., M. Mollaghasemi, and D. Nazzal 2001. Scheduling setup changing at bottleneck facilities in semiconductor manufacturing. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 1208-1214. Arlington, VA, USA.

El-Mikawi, M. 1996. A methodology for evaluation of the use of advanced composites in structural civil engineering applications, *Composites: Part B*, vol. 27, no. 3-4, 203-215.

Eldabi, T. and R. J. Paul 2001. A proposed approach for modelling healthcare systems for understanding. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 1412-1420. Arlington, VA, USA.

Eneyo, E. S. and G. P. Pannirselvam 1998. The use of simulation in facility layout design: A practical consulting experience. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 1527-1532. Washington DC, USA.

Fahmy, H. M. A. 2001. Reliability evaluation in distributed computing environments using AHP, *Computer Networks*, vol. 37, no. 5-6, 597-615.

Ferscha, A. and A. Tripathi 1994. Parallel and distributed simulation of discrete event systems, *Technical Report*, Department of Computer Science, University of Maryland, USA, CS-TR-3336.

Ferscha, A. 1995. Parallel and distributed simulation of discrete event systems, in *Parallel and Distributed Computing Hand Book*, First edn, McGraw-Hill, 1003-1041.

Ferscha, A., J. Johnson, and S. J. Turner 2001. Distributed simulation performance data mining, *Future Generation Computer Systems*, vol. 18, 157-174.

Firat, C. 2000. Conceptual modelling and conceptual analysis in HLA. In *Proceedings of 2000 Simulation Interoperability Workshops.* 17-22.

Fishwick, P. A. 1994. Simulation model design. In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 173-175. Orlando, FL, USA.

Forman, E. H. 2001. Decision by Objectives, http://mdm.gwu.edu/forman/ ,

Fujimoto, R. M. 1990. Parallel discrete event simulation, *Communications of the ACM*, vol. 33, no. 10, 30-53.

Fujimoto, R. M.1993. Parallel and distributed discrete event simulation: Algorithms and applications. In *Proceedings of 1993 Winter Simulation Conference*, ed. E. C. Russell, W. E. Biles, G. W. Evans, and M. Mollaghasemi. 106-114. Los Angeles, CA, USA.

Fujimoto, R. M. 1993a. Parallel discrete event simulation: Will the field survive?, *ORSA Journal on Computing*, vol. 5, no. 3, 213-230.

Fujimoto, R. M. 1993b. Future directions in parallel simulation research, *ORSA Journal on Computing*, vol. 5, no. 3, 245-248.

Fujimoto, R. M. 1998. Parallel and distributed simulation, in *Handbook of Simulation*, first edn, ed. J. Banks, 429-464, John Wiley & Sons.

Fujimoto, R. M. 1999. Parallel and distributed simulation. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nembhard. 121-132. Phoenix, AZ, USA.

Fujimoto, R. M. 2000. *Parallel and Distributed Simulation Systems*, First edn, John Wiley.

Fujimoto, R. M. 2001. Parallel and distributed simulation systems. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 147-157. Arlington, VA, USA.

Gan, B. P., L. Liu, S. Jain, S. J. Turner, W. Cai, and W.-J. Hsu 2000. Distributed supply chain simulation across enterprise boundaries. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 1245-1251. Orlando, FL, USA.

Gerogiannis, V. C., A. D. Kameas, and P. E. Pintelas 1998. Comparative study and categorization of high-level Petri nets, *Journal of Systems and Software*, vol. 43, no. 2, 133-160.

Gile, M. R. and F. DiCesare, 2001, Toward distributed simulation of complex discrete event systems represented by colored Petri nets: A review, http://citeseer.nj.nec.com/11819.html , Goldsman, D. and G. Tokol. 2000. Output analysis procedures for computer simulation. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 39-45. Orlando, FL, USA.

Haddix, F. 2001. Conceptual modelling revisited: A developmental model approach for modelling & simulation. In *Proceedings of 2001 Simulation Interoperability Workshops,*

Hamilton, J. A. Jr., Nash, D. A., and Pooch, U. W. 1997. *Distributed Simulation,* CRC Press, New York.

Hamnes, D. O. and A. Tripathi 1994. Investigation in adaptive distributed simulation. In *Proceedings of 1994 Workshop on Parallel and Distributed Simulation.* 20-23. Edinburgh, Scotland, United Kingdom.

Hao, F., K. Wilson, R. M. Fujimoto, and E. Zegura 1996. Logical process size in parallel simulation. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 645-652. Coronado, CA, USA.

Hendrickson, B. and T. G. Kolda 2000. Graph partitioning models for parallel computing, *Parallel Computing*, vol. 26, no. 12, 1519-1534.

Hibino, H., Y. Fukuda, Y. Yura, K. Mitsuyuki, and K. Kaneda 2002. Manufacturing adapter of distributed simulation systems using HLA. In *Proceedings of 2002 Winter Simulation Conference*, ed. J. L. Snowdon, J. M. Charnes, E. Yucesan, and C. Chen. 1099-1107. San Diego, CA , USA.

Ikonen, J. and J. Porras 1998. Applying distributed simulation. In *Proceedings of 1998 European Simulation Symposium.*

Ingalls, R. G. 2002. Introduction to simulation. In *Proceedings of 2002 Winter Simulation Conference*, ed. J. L. Snowdon, J. M. Charnes, E. Yucesan, and C. Chen. 7-16. San Diego, CA , USA.

Jain, S. 1999. Simulation in the next millennium. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nembhard. 1478-1484. Phoenix, AZ, USA.

Jefferson, D. R. 1985. Virtual time, *ACM Transactions on Programming Languages and Systems*, vol. 7, no. 3, 404-425.

Johnson, B. C. 1991, *A distributed computing environment framework: An OFS perspective*, Open Software Foundation.

Kadar, B., L. Monostori, and E. Szelke 1998. An object-oriented framework for developing distributed manufacturing architecture, *Journal of Intelligent Manufacturing*, vol. 9, no. 2, 173-179.

Karacal, S. C. 1998. A novel approach to simulation modelling, *Computers and Industrial Engineering*, vol. 34, no. 3, 573-587.

Kateel, G., M. Kamath, and D. Pratt 1996. An overview of CIM enterprise modelling methodologies. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 1000-1007. Coronado, CA, USA.

Kelton, W. D. 2000. Experimental design for simulations. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 32-38. Orlando, FL, USA.

Kienbaum, G. and R. J. Paul 1994. H-ACD: Hierarchical activity cycle diagrams for object oriented simulation modelling. In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 600-610. Orlando, FL, USA.

Kim, K. H., Y. R. Seong, T. G. Kim, and K. H. Park 1997. Ordering of simultaneous events in distributed DEVS simulation, *Simulation: Practice and Theory*, vol. 5, no. 3, 253-268.

Kim, K. H., T. G. Kim, and K. H. Park 1998. Hierarchical partitioning algorithm for optimistic distributed simulation of DEVS models, *Journal of Systems Architecture*, vol. 44, no. 6-7, 433-455.

Kiran, A. S. 1998. Hierarchical modelling: A simulation based application. In *proceedings of the IEEE International Conference on Systems, Man and Cybernetics*. 3079-3089.

Kirchner, T. B. 1997. Distributed processing applied to ecological modelling, *Simulation: Practice and Theory*, vol. 5, no. 1, 35-47.

Koh, K.-H., R. De Souza, and N.-C. Ho 1996. Multi-processor distributed simulation for job-shop scheduling: boon or bane?, *International Journal of Computer Integrated Manufacturing*, vol. 9, no. 6, 432-442.

Korn, S., G. R. Burns, and D. K. Harrison 1999. The application of multiparadigm simulation techniques to manufacturing processes, *International Journal of Advanced Manufacturing Technology*, vol. 15, 869-875.

Law, A. M. & Kelton, W. D. 1991. *Simulation Modelling and Analysis* McGraw-Hill, New York.

Law, A. M. and M. G. McComas 1998. Simulation of manufacturing systems. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 49-52. Washington DC, USA.

Law, A. M. and M. G. McComas 1999. Simulation of manufacturing systems. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard. 56-59. Phoenix, AZ, USA.

Law, A. M. and M. G. McComas 2001. How to build valid and credible simulation models. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 22-29. Arlington, VA, USA.

Leemis, L. 2001. Input modelling techniques for discrete-event simulations. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 62-73. Arlington, VA, USA.

Leitao, P. and F. Restivo 2000. A framework for distributed manufacturing application. In *Proceedings of 2000 Advanced Summer Institute (ASI) Conference*. 1-7. Bordeaux, France.

Lendermann, P., B. P. Gan, and L. F. McGinnis 2001. Distributed simulation with incorporated APS procedures for high-fidelity supply chain optimization. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 1138-1145. Arlington, VA, USA.

Leung, J. W. and K. K. Lai 1997. A structured methodology to build discrete-event simulation models, *Asia-Pacific Journal of Operational Research*, vol. 14, no. 1, 19-37.

Lilegdon, W. R. 1996. Simulation works: A panel discussion. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 1337-1340. Coronado, CA, USA.

Lin, Y.-B. 1993. Special issue on parallel discrete event simulation, *Journal of Parallel and Distributed Computing*, vol. 18, no. 4, 391-394.

Lin, Y.-B. and P. A. Fishwick 1996. Asynchronous parallel discrete event simulation, *IEEE Transaction of the Systems, Man and Cybernetics*, vol. 24, no. 6, 397-412.

Lin, Y.-B. 2000. Design issues for optimistic distributed discrete event simulation, *Journal of Information Science and Engineering*, vol. 16, 243-269.

Linn, R. J., C.-S. Chen, and J. A. Lozan 2002. Development of distributed simulation model for the Transporter entity in a supply chain process. In *Proceedings of 2002 Winter Simulation Conference*, ed. J. L. Snowdon, J. M. Charnes, E. Yucesan, and C. Chen. 1319-1326. San Diego, CA , USA.

Low, Y.-H., C.-C. Lim, W. Cai, S.-Y. Huang, S. Jain, and S. J. Turner 1999. Survey of languages and runtime libraries for parallel discrete-event simulation, *Simulation*, vol. 72, no. 3, 170-186.

Luksch, P., 1995, Parallel logic simulation on distributed memory multiprocessors: classification and evaluation of different approaches, http://citeseer.nj.nec.com/luksch95parallel.html .

Luksch, P. 2002, A test environment for the evaluation of different approaches to distributed logic simulation, http://citeseer.nj.nec.com/379268.html.

Luna, J. J. 1992. Hierarchical, modular concepts applied to an object-oriented simulation model development environment. In *Proceedings of 1992 Winter Simulation Conference*, ed. R. C. Crain, J. R. Wilson, J. J. Swain, and D. Glodsman. 694-699. Arlington, VA, USA.

Maria, A. 1997. Introduction to modelling and simulation. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 7-13. Atlanta, GA, USA.

Mastaglio, T. W. 1999. Enterprise simulations: Theoretical foundations and a practical perspective. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard. 1485-1489. Phoenix, AZ, USA.

McLean, C. and F. Riddick 2000. The IMS mission architecture for distributed manufacturing simulation. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 1539-1548. Orlando, FL, USA.

McLean, C. and S. Leong 2001. The role of simulation in strategic manufacturing. In *Proceedings of 2001 International Working Conference on Strategic Manufacturing*. 239-250. Aalborg, Denmark.

McLean, C. and G. Shao 2001. Simulation of shipbuilding operations. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 870-876. Arlington, VA, USA.

Mehl, H. and S. Hammes 1993. Shared variables in distributed simulation. In *Proceedings of 1993 Workshop on Parallel and Distributed Simulation*. 68-75. San Diego, California, United States.

Mielke, R. R. 1999. Applications for enterprise simulation. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard. 1490-1495. Phoenix, AZ, USA.

Minton, G., 2003, programming with CORBA, http://www.blackmagic.com/people/gabe/prog-with-corba.html.

Misra, J. 1986. Distributed discrete-event simulation, *Computing Surveys*, vol. 18, no. 1, 39-65.

Nakayama, M. K. 2002. Simulation output analysis. In *Proceedings of 2002 Winter Simulation Conference*, ed. J. L. Snowdon, J. M. Charnes, E. Yucesan, and C. Chen. 23-33. San Diego, CA , USA.

Nandy, B. and W. M. Loucks 1992. An algorithm for partitioning and mapping conservative parallel simulation onto multicomputers. In *Proceeding of the 1992 SCS Multiconference on Parallel and Distributed Simulation*,

Nandy, B. and W. M. Loucks 1993. On a parallel partitioning technique for use with conservative parallel simulation. In *Proceedings of 1993 Workshop on Parallel and Distributed Simulation*. 43-51. San Diego, California, United States.

Nicol, D. M. 1993. The cost of conservative synchronisation in parallel discrete event simulations, *Journal of the Association for Computing Machinery*, vol. 40, no. 2, 304-333.

Nicol, D. M. and R. M. Fujimoto 1994. Parallel simulation today, *Annals of Operations Research*, vol. 53, 1-34.

Nicol, D. M. and P. Heidelberger 1995. On extending parallelism to serial simulators. In *Proceedings of 1995 Workshop on Parallel and Distributed Simulation.* 60-67. New York, United States.

Nicol, D. M. 1996. Principles of conservative parallel simulation. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 128-135. Coronado, CA, USA.

Nordgren, W. B. 1995. Steps for proper simulation project management. In *Proceedings of 1995 Winter Simulation Conference* , ed. W. R. Lilegdon, D. Glodsman, C. Alexopoulos, and K. Kang. 68-73. Arlington, VA, USA.

Nutt, G. J. 1990. Distributed simulation design alternatives. In *Proceeding of the SCS Multiconference on Distributed Simulation.*

Odhabi, H. I., R. J. Paul, and R. D. Macredie 1997. The four phase method for modelling complex systems. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 510-517. Atlanta, GA, USA.

Odhabi, H. I., R. J. Paul, and R. D. Macredie 1998. Making simulation more accessible in manufacturing systems through a 'four phase' approach. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 1069-1075. Washington DC, USA.

OMG, 2002, CORBA overview, http://www.infosys.tuwien.ac.at/research/corba /OMG/arch2.htm#44686 , Last accessed in 2002.

Ossadnik, W. and O. Lange 1999. AHP - based evaluation of AHP - software, *European Journal of Operational Research*, vol. 118, no. 3, 578-588.

Overeinder, B., B. Hertzberger, and P. Sloot 1991. Parallel discrete event simulation. In *The Workshop on Computer systems, Faculty of Electrical Engineering, Eindhoven University, The Netherlands*, ed. W. J. Withagen. 19-30.

Pace, D. K. 1999. Conceptual model descriptions. In *Proceedings of 1999 Simulation Interoperability Workshops.*

Pace, D. K. 2000. Ideas about simulation conceptual model development, *John Hopkins Applied Technical Digest*, vol. 21, no. 3, 327-336.

Page, E. H. and R. E. Nance 1994. Parallel discrete event simulation: A modelling methodological perspective. In *Proceedings of 1994 Workshop on Parallel and Distributed Simulation.* 88-93. Edinburgh, Scotland, United Kingdom.

Page, E. H. 1999. Panel: Strategic directions in simulation research. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nembhard. 1509-1520. Phoenix, AZ, USA.

Pancake, C. M. 1996. Is parallelism for you?, *IEEE Computational Science and Engineering*, vol. Summer, 1996, 18-37.

Panda, D. K. and L. M. Ni 1997. Special issue on workstation clusters and network-based computing, *Journal of Parallel and Distributed Computing*, vol. 40, no. 1, 1-3.

Pandya, K. V. 1995. Review of modelling techniques and tools for decision making in manufacturing management. In *IEE Proceedings of Science, Measures and Technology.*

Pandya, K. V., A. Karlsson, S. Sega, and A. Carrie 1997. Towards the manufacturing enterprises of the future, *International Journal of Operations and Production Management*, vol. 17, no. 5, 502-521.

Pasquini, R. and V. Rego 1998. Efficient process interaction with threads in parallel discrete event simulation. In *Proceedings of 1999 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 451-458. Washington DC, USA.

Peng, C. and F. F. Chen 1996. Parallel discrete event simulation of manufacturing systems: A technology survey, *Computers and Industrial Engineering*, vol. 31, no. 1/2, 327-330.

Perez, J. 1995. Some comments on Saaty's AHP, *Management Science*, vol. 41, no. 6, 1091-1095.

Peterson, G. D. and J. C. Willis 1999. High-performance hardware description language simulation: Modelling issues and recommended practices, *Transactions of the Society for Computer Simulation International*, vol. 16, no. 1, 6-15.

Pflughoeft, K. A. and K. Manur 1994. Multi-layered activity cycle diagrams and their conversions into activity-based simulation code. In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 595-599. Orlando, FL, USA.

Pham, C. D., H. Brunst, and S. Fdida 1998. How can we study large and complex systems. In *Proceedings of IEEE Annual Simulation Symposium.*

Pidd, M. 1994. An introduction to computer simulation. In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 7-14. Orlando, FL, USA.

Pidd, M. 1996. Five simple principles of modelling. In *Proceedings of 1996 Winter Simulation Conference*, ed. J. M. Charnes, D. J. Morrice, D. T. Brunner, and J. J. Swain. 721-728. Coronado, CA, USA.

Pidd, M. and B. R. Castro 1998. Hierarchical modular modelling in discrete simulation. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 383-390. Washington DC, USA.

Pidd, M., N. Oses, and R. J. Brooks 1999. Component-based simulation on the web? In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard 1438-1444. Phoenix, AZ, USA.

Pooley, R. J. 1991. Aggregation and hierarchical modelling (Part III), *Transactions of the Society for Computer Simulation International*, vol. 8, no. 1, 33-41.

Porras, J., V. Hara, J. Harju, and J. Ikonen 1997. Improving the performance of the Chandy-Misra parallel simulation algorithm in a distributed workstation environment. In *Proceedings of 1997 Summer Computer Simulation Conference*.

Poyhonen, M. and R. P. Hamalainen 2001. On the convergence of multiattribute weighting methods, *European Journal of Operational Research*, vol. 129, no. 3, 569-585.

Randell, L. G., L. G. Holst, and G. S. Bolmsjo 1999. Incremental system development of large discrete-event simulation models. In *Proceedings of 1999 Winter Simulation Conference*, ed. D. T. Sturrock, G. W. Evans, P. A. Farrington, and H. B. Nemhard. 561-458. Phoenix, AZ, USA.

Resenburg, A. V. and N. Zwemstra 1995. Implementing IDEF techniques as simulation modelling specifications, *Computers and Industrial Engineering*, vol. 29, no. 1-4, 467-471.

Reynolds, P. F. Jr. 1988. A spectrum of options for parallel simulation. In *Proceedings of 1988 Winter Simulation Conference*, ed. P. L. Haigh, J. C. Comfort, and M. A. Abrams. 325-332. San Diego, CA, USA.

Righter, R. and J. C. Walrand 1989. Distributed simulation of discrete event systems, *Proceedings of IEEE*, vol. 77, no. 1, 99-113.

Robinson, S. 1994. *Successful Simulation: A Practical Approach to Simulation Projects* McGraw-Hill.

Saad, S. M. 2003. The reconfiguration issues in manufacturing systems, *Journal of Materials Processing Technology* no. 138, 277-283.

Sadoun, B. 2000. Applied system simulation: a review study, *Information Sciences*, vol. 124, no. 1-4, 173-192.

Sadowski, R. P. 1991. Avoiding the problems and pitfalls in simulation. In *Proceedings of 1991 Winter Simulation Conference*, ed. W. D. Kelton, G. M. Clark, and B. L. Nelson. 48-55. Phoenix, AZ, USA.

Sanchez, S. M. 2001. ABC's of output analysis. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 30-38. Arlington, VA, USA.

Sanchez, V., A. Bautista, and F. Tirado 1996. Deblocking event algorithm: a new approach to conservative parallel discrete event simulation. In *Proceedings of 1996 Euromicro Workshop on Parallel and Distributed processing.*

Sargent, R. G., J. H. Mize, d. H. Withers, and B. P. Zeigler 1993. Hierarchical modelling for discrete event simulation (panel). In *Proceedings of 1993 Winter Simulation Conference*, ed. E. C. Russell, W. E. Biles, G. W. Evans, and M. Mollaghasemi. 569-572. Los Angeles, CA, USA.

Sargent, R. G. 1994. Verification and validation of simulation models. In *Proceedings of 1994 Winter Simulation Conference*, ed. D. A. Sadowski, A. F. Seila, J. D. Tew, and S. Manivannan. 77-87. Orlando, FL, USA.

Sargent, R. G. 2000. Verification, validation, and accreditation of simulation models. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 50-59. Orlando, FL, USA.

Sargent, R. G. 2001. Some approaches and paradigms for verifying and validating simulation models. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 106-114. Arlington, VA, USA.

Sawhney, A., O. Abudayyeh, and A. Monga 1999. Modelling and analysis of a mail processing plant using Petri nets, *Advances in Engineering Software*, vol. 30, 543-549.

Sawhney, A. 2000. An integrated modelling methodology for simulation of large and complex systems, *International Journal of Modelling and Simulation*, vol. 20, no. 1, 1-11.

Schreiber, R., 1995, Middleware demystified, Datamation, April, 41-45,

Schruben, L. 1983. Simulation modelling with event graphs, *Communications of the ACM*, vol. 26, no. 11, 957-967.

Shannon, R. E. 1992. Introduction to simulation. In *Proceedings of 1992 Winter Simulation Conference*, ed. R. C. Crain, J. R. Wilson, J. J. Swain, and D. Glodsman. 65-73. Arlington, VA, USA.

Shannon, R. E. 1998. Introduction to the art and science of simulation. In *Proceedings of 1998 Winter Simulation Conference*, ed. J. S. Carson, M. S. Manivannan, D. J. Medeiros, and E. F. Watson. 7-14. Washington DC, USA.

Shen, W. and D. H. Norrie 1998. An agent-based approach for manufacturing enterprise integration and supply chain management. In *Proceedings of 1998 International Conference on the practical Applications of Agents and Multi-agent Systems,*

Sherif, Y. S. 1998. The design, analysis, and evaluation of simulation experiments, *International Journal of Modelling and Simulation*, vol. 18, no. 4, 290-297.

Shi, J. 1997. A conceptual activity cycle-based simulation modelling method. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 1127-1133. Atlanta, GA, USA.

Silva, L., A. L. Ramos, and P. M. Vilarinho 2000. Using simulation for manufacturing process reengineering - A practical case study. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 1322-1328. Orlando, FL, USA.

Sirinivasan, K. and S. Jayaraman 1997. Integration of simulation with enterprise models. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 1352-1356. Atlanta, GA, USA.

Sohl, B. A., 2002, Distributed simulation: Concepts and application, http://www.cecs/csulb/edu/~sohl/distributedsim/dissim/dissim-bogy.html,

Solcany, V., R. Skultety, and J. Safarik 1995. Simulation model decomposition in conservative parallel discrete event simulation. In *Proceedings of the 1995 European Simulation Multiconference.*

Steuer, R. E. and P. Na 2003. Multiple criteria decision making combined with finance: A categorised bibliographic study, *European Journal of Operational Research*, vol. 150, no. 1, 496-515.

Sudra, R., S. J. E. Taylor, and T. Janahan 2000. Distributed supply chain simulation in GRIDS. In *Proceedings of 2000 Winter Simulation Conference*, ed. P. A. Fishwick, K. Kang, J. A. Joines, and R. H. Barton. 356-361. Orlando, FL, USA.

Szynkiewicz, E. N. 2000. Parallel and distributed simulation; Methodology, tools and applications, in *Advances in Multi-agent Systems*, ed. R. Schaefer and S. Sedziwy.

Takus, D. A. and D. M. Profozich 1997. Arena software tutorial. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 541-544. Atlanta, GA, USA.

Tam, M. C. Y. and V. M. R. Tummala 2001. An application of the AHP in vendor selection of a telecommunication system, *Omega: The International Journal of management Science*, vol. 29, no. 2, 171-182.

Taylor, S. J. E. 1998. Parallel and distributed simulation, *Journal of Systems Architecture*, vol. 44, no. 6-7, 393-394.

Taylor, S. J. E., R. Sudra, T. Janahan, G. Tan, and J. Ladbrook 2001. Towards COTS distributed simulation using GRIDS. In *Proceedings of 2001 Winter Simulation Conference*, ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 1372-1379. Arlington, VA, USA.

Taylor, S. J. E. 2002. Distributed simulation and Industry: Potentials and pitfalls (Panel discussion). In *Proceedings of 2002 Winter Simulation Conference*, ed. J. L. Snowdon, J. M. Charnes, E. Yucesan, and C. Chen. 688-694. San Diego, CA, USA.

Theodoropoulos, G. K. 1995, *Strategies for the modelling and simulation of asynchronous computer architectures,* University of Manchester, United Knigdom.

Thesen, A. and L. A. Travis 1990. Introduction to simulation. In *Proceeding of Winter 1990 Simulation Conference*, ed. O. Balci, R. P. Sadowski, and R. E. Nance. 14-21. New Orleans, LA , USA.

Turner, S. J. 1998. Models of computation for parallel discrete event simulation, *Journal of System Architecture*, vol. 44, no. 6-7, 395-409.

Vee, V. Y. & W. J. Hsu 1999, *Parallel discrete event simulation*, Centre for Advanced Information Systems, Nanyang Technological University, Singapore.

Venkateswaran, J., M. Y. K. Jafferali, and Y. J. Son 2001. Distributed simulation: An enabling technology for the evaluation of virtual enterprises. In *Proceedings of 2001 Winter Simulation Conference* , ed. M. Rohrer, D. Medeiros, B. A. Peters, and J. Smith. 856-862. Arlington, VA, USA.

Vojnar, T. 1997. Various Kinds of Petri Nets in Simulation and Modelling. In *Proceedings of 1997 Spring International Conference on Modelling and Simulation of Systems MOSIS'97.*

Vondrak, C. and R. Beach 1997. Distributed Computing Environment, http://www.sei.cmu.edu/str/descriptions/dce.html, Last accessed in June 2003.

Wallnau, K. 1997. Common object request broker architecture, http://www.sei.cmu.edu/str/descriptions/corba.html, Last accessed in June 2003.

Whitman, L., B. Huff, and A. Presley 1997. Structured models and dynamic systems analysis: The integration of the IDEF0/IDEF3 modelling methods and discrete event simulation. In *Proceedings of 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 518-524. Atlanta, GA, USA.

Wild, R. H. and J. J. Jr. Pignatiello 1991. An experimental design strategy for designing robust systems using discrete-event simulation, *Simulation*, vol. 57, no. 6, 358-368.

Wilson, J. R. 1997. Modelling dependencies in stochastic simulation inputs. In *Proceeding of the 1997 Winter Simulation Conference*, ed. D. H. Withers, B. L. Nelson, S. Andradottir, and K. J. Healy. 47-52. Atlanta, GA, USA.

Yang, C. and J. B. Huang 2000. A decision model for IS outsourcing, *International Journal of Information Management*, vol. 20, no. 3, 225-239.

Yapa, S. 2003. *A structured approach to rapid simulation model development.* An on going research leading to a PhD.

Yusuff, R. M., K. P. Yee, and M. S. J. Hashmi 2001. A preliminary study on the potential use of the analytic hierarchical process (AHP) to predict advanced manufacturing technology (ATM) implementation, *Robotics and Computer Integrated Manufacturing*, vol. 17, no. 5, 421-427.

Zahedi, F. 1986. The analytic hierarchy process - A survey of the method and its applications, *Interfaces*, vol. 16, no. 4, 96-108.

Zanakis, S. H., A. Solomon, N. Wishart, and S. Dublish 1998. Multi-atribute decision making: A simulation comparison of select methods, *European Journal of Operational Research*, vol. 107, no. 3, 507-529.

Zeigler, B. P. 1986. Hierarchical modular modelling/ knowledge representation. In *Proceedings of 1986 Winter Simulation Conference*, ed. J. O. Henriksen, S. D. Roberts, and J. R. Wilson. 129-137. Washington, DC, USA.

Zeigler, B. P. 1987. Hierachical, modular discrete-event modeling in an object-oriented environment, *Simulation*, vol. 49, no. 5, 219-330.

Zupancic, B. 1998. Modular hierarchical modelling with SIMCOS language, *Mathematics and Computers in Simulation*, vol. 46, no. 1, 67-76.

Appendix 1 – Application program interface (API) for model B

```
Option Explicit

Dim BPaused As Integer
Dim sqQueue As MSMQQueue
Dim pqQueue As MSMQQueue
Public WithEvents sqEvent As MSMQEvent
Public WithEvents pqEvent As MSMQEvent

Private Sub ModelLogic_RunBeginSimulation()
  'receives parts from A
  Dim qinfo As New MSMQQueueInfo
  qinfo.PathName = ".\private$\bpq"
  Set pqQueue = qinfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)
  Set pqEvent = New MSMQEvent
  pqQueue.EnableNotification pqEvent
  Call ThisDocument.Model.Pause
End Sub

Private Sub ModelLogic_RunBeginReplication()
  'Processes times
  BPaused = 0
  Dim sqInfo As New MSMQQueueInfo
  sqInfo.PathName = ".\private$\bsq"
  Set sqQueue = sqInfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)
  Set sqEvent = New MSMQEvent
  sqQueue.EnableNotification sqEvent
End Sub

Private Sub ModelLogic_RunEndSimulation()
  'collecting total times, MC utilisations and sending to
TPU
  Dim ary(8) As String
  Dim Aarstr As String
  Dim line As String
  Dim i As Integer
  Dim j As Integer
  Dim k As Integer

  Open "D:\apr\modified case study\Firm B_sync_2.out" For
                                                Input As #1

  For i = 1 To 200
    If EOF(1) Then
      Exit For
    End If

    Input #1, line

    If Trim(Mid(line, 1, 22)) = "Entity 2.TotalTime" Then
    ary(1) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 22)) = "MC1.Utilization" Then
    ary(2) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 22)) = "MC2.Utilization" Then
    ary(3) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 22)) = "MC3.Utilization" Then
    ary(4) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 22)) = "MC4.Utilization" Then
    ary(5) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 22)) = "MC5.Utilization" Then
    ary(6) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 22)) = "MC6.Utilization" Then
    ary(7) = Trim(Mid(line, 20, 13))
  End If

  If Trim(Mid(line, 1, 25)) = "Entity 2.NumberOut" Then
    ary(8) = Trim(Mid(line, 26, 15))
  End If

Next i

Close #1

Aarstr = CStr(ary(1))
For j = 2 To 8
  Aarstr = Aarstr & ":" & CStr(ary(j))
Next j
Aarstr = Aarstr & ":"

Dim paInfo As New MSMQQueueInfo
Dim paDest As MSMQQueue
Dim pamsgSend As New MSMQMessage
pamsgSend.Label = "PERFORMANCE"
pamsgSend.Body = Aarstr
paInfo.FormatName = "DIRECT = OS:ENG-4112-07\private$\btq"
Set paDest = paInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
pamsgSend.Send paDest
paDest.Close

'***assingning values to "sample measures for Firm B"
UserForm1.PTZ = ary(1)
UserForm1.MUMC1 = ary(2)
UserForm1.MUMC2 = ary(3)
```

```
  UserForm1.MUMC3 = ary(4)
  UserForm1.MUMC4 = ary(5)
  UserForm1.MUMC5 = ary(6)
  UserForm1.MUMC6 = ary(7)
  UserForm1.OutZ = ary(8)
  UserForm2.CommandButton1.Enabled = False
  UserForm2.Show
End Sub

Private Sub ModelLogic_RunPause()
  'processes Pause and restart
  Dim sqInfo As New MSMQQueueInfo
  sqInfo.PathName = ".\private$\bsq"
  Set sqQueue = sqInfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)
  Set sqEvent = New MSMQEvent
  sqQueue.EnableNotification sqEvent
End Sub

Private Sub ModelLogic_RunResume()
  'processes Pause and restart
  Dim sqInfo As New MSMQQueueInfo
  sqInfo.PathName = ".\private$\bsq"
  Set sqQueue = sqInfo.Open(MQ_RECEIVE_ACCESS, MQ_DENY_NONE)
  Set sqEvent = New MSMQEvent
  sqQueue.EnableNotification sqEvent
End Sub

Private Sub pqEvent_Arrived(ByVal Queue As Object, ByVal
Cursor As Long)
  Dim vEntityIndex As Long
  Dim vPictureIndex As Long
  Dim sTime As Variant
  Dim cTime As Double
  Dim dTime As Double
  Dim aEResults(6) As String
  Dim vEresults As String
  Dim i As Integer
  Dim j As Integer
  Dim k As Integer
  Dim l As Integer
  Dim bQueue As MSMQQueue
  Set bQueue = Queue
  Dim qMsg As MSMQMessage
  Set qMsg = New MSMQMessage
  Set qMsg = bQueue.Receive(, , , 0)

  If qMsg.Label = "EResults" Then
    vEresults = qMsg.Body
    'MsgBox "vEresults"
    'MsgBox vEresults
    i = 1
    j = 1
    For k = 1 To Len(vEresults)
      If Mid(vEresults, k, 1) = ":" Then
        aEResults(i) = Mid(vEresults, j, k - j)
        'MsgBox i
        'MsgBox aEResults(i)
        i = i + 1
        j = k + 1
      End If
    Next k

    UserForm1.EPTX.Caption = aEResults(1)
    UserForm1.EPTY.Caption = aEResults(2)
    UserForm1.EPTZ.Caption = aEResults(3)
    UserForm1.EPTXYZ.Caption = aEResults(4)
    UserForm1.CTXYZ.Caption = aEResults(5)
    UserForm1.EOUTXYZ.Caption = aEResults(6)
    UserForm2.CommandButton1.Enabled = True

  Else

    vPictureIndex =
       ThisDocument.Model.SIMAN.SymbolNumber("Picture.yellow
       page")

    For i = 1 To 1
      vEntityIndex = ThisDocument.Model.SIMAN.EntityCreate
      Call
       ThisDocument.Model.SIMAN.EntitySetPicture(vEntityInde
       x, vPictureIndex)
      Call
       ThisDocument.Model.SIMAN.EntitySendToBlockLabel(vEnti
       tyIndex, 0, "AInput")
    Next i

  End If

  bQueue.EnableNotification pqEvent

End Sub

Private Sub sqEvent_Arrived(ByVal Queue As Object, ByVal
Cursor As Long)
  Dim lIndex As Long
  Dim pIndex As Long
  Dim sPTime As Double
  Dim sSATime As Double
  Dim sSBTime As Double
  Dim sSCTime As Double
  Dim EResults(6) As String
    Dim sprQueue As MSMQQueue
  Set sprQueue = Queue
  Dim sprqMsg As MSMQMessage
  Set sprqMsg = sprQueue.Receive(, , , 0)

  If sprqMsg.Label = "START" Then
    Call ThisDocument.Model.Go
  End If
```

```
If sprqMsg.Label = "STOP" Then
  Call ThisDocument.Model.End
End If

'To resume A
If sprqMsg.Label = "PTA" Then
  sSBTime = ThisDocument.Model.SIMAN.RunCurrentTime
  sSATime = CDbl(sprqMsg.Body)
  sPTime = sSATime - sSBTime
  If sPTime < 0.5 Then
    sPTime = 0
    lIndex = ThisDocument.Model.SIMAN.EntityCreate
    pIndex =
        ThisDocument.Model.SIMAN.SymbolNumber("Picture.Tr
        uck")
    Call ThisDocument.Model.SIMAN.EntitySetPicture(lIndex,
        pIndex)
    Call
        ThisDocument.Model.SIMAN.EntitySendToBlockLabel(l
        Index, sPTime, "RAblock")
  Else
    lIndex = ThisDocument.Model.SIMAN.EntityCreate
    pIndex =
        ThisDocument.Model.SIMAN.SymbolNumber("Picture.va
        n")
    Call ThisDocument.Model.SIMAN.EntitySetPicture(lIndex,
        pIndex)
    Call
        ThisDocument.Model.SIMAN.EntitySendToBlockLabel(l
        Index, sPTime, "RAblock")
  End If
End If

 'To resume C
If sprqMsg.Label = "PTC" Then
  sSBTime = ThisDocument.Model.SIMAN.RunCurrentTime
  sSCTime = CDbl(sprqMsg.Body)
  sPTime = sSCTime - sSBTime
  If sPTime < 0.5 Then
    sPTime = 0
    lIndex = ThisDocument.Model.SIMAN.EntityCreate
    pIndex =
        ThisDocument.Model.SIMAN.SymbolNumber("Picture.va
        n")
    Call ThisDocument.Model.SIMAN.EntitySetPicture(lIndex,
        pIndex)
    Call
        ThisDocument.Model.SIMAN.EntitySendToBlockLabel(l
        Index, sPTime, "RCblock")
  Else
    lIndex = ThisDocument.Model.SIMAN.EntityCreate
    pIndex =
        ThisDocument.Model.SIMAN.SymbolNumber("Picture.va
        n")
```

```
      Call ThisDocument.Model.SIMAN.EntitySetPicture(lIndex,
          pIndex)
      Call
          ThisDocument.Model.SIMAN.EntitySendToBlockLabel(l
          Index, sPTime, "RCblock")
    End If
  End If

  'sending time to TPU
  If sprqMsg.Label = "RT" Then
    Dim tbInfo As New MSMQQueueInfo
    Dim tbDest As MSMQQueue
    Dim tbmsgSend As New MSMQMessage
    tbmsgSend.Label = "T"
    tbmsgSend.Body = ThisDocument.Model.SIMAN.RunCurrentTime
    tbInfo.FormatName = "DIRECT = OS:ENG-4112-
07\private$\btq"
    Set tbDest = tbInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
    tbmsgSend.Send tbDest
    tbDest.Close
  End If

  'pausing B
  If sprqMsg.Label = "P" And BPaused = 0 Then

    Dim FName As String

    If sprqMsg.Body = "A" Then
      FName = "DIRECT = OS:ENG-4112-07\private$\asq"
    End If

    If sprqMsg.Body = "C" Then
      'FName = "DIRECT = OS:ENG-4130-12\private$\csq"
      FName = "DIRECT = OS:ENG-4112-09-od\private$\csq"
    End If

    Dim ptInfo As New MSMQQueueInfo
    Dim ptDest As MSMQQueue
    Dim ptmsgSend As New MSMQMessage
    ptmsgSend.Label = "PTB"
    ptmsgSend.Body = ThisDocument.Model.SIMAN.RunCurrentTime
    ptInfo.FormatName = FName
    Set ptDest = ptInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
    ptmsgSend.Send ptDest
    ptDest.Close
    BPaused = 1
    Call ThisDocument.Model.Pause
  End If


  'resuming B
  If sprqMsg.Label = "R" Then
    'updating TPU
    Dim tpuqInfo As New MSMQQueueInfo
    Dim tpuqDest As MSMQQueue
    Dim tpumsgSend As New MSMQMessage
    tpumsgSend.Label = "R"
    tpumsgSend.Body =
ThisDocument.Model.SIMAN.RunCurrentTime
    tpuqInfo.FormatName = "DIRECT = OS:ENG-4112-
07\private$\btq"
    Set tpuqDest = tpuqInfo.Open(MQ_SEND_ACCESS,
MQ_DENY_NONE)
    tpumsgSend.Send tpuqDest
    tpuqDest.Close
    BPaused = 0
    Call ThisDocument.Model.Go
  End If

  sprQueue.EnableNotification sqEvent
End Sub

Private Sub VBA_Block_1_Fire()
  'passing parts to C
  Dim cqInfo As New MSMQQueueInfo
  Set cqInfo = New MSMQQueueInfo
  Dim cTime As Double
  cTime = ThisDocument.Model.SIMAN.RunCurrentTime
  'cqInfo.FormatName = "DIRECT = OS:ENG-4130-
12\private$\cpq"
  cqInfo.FormatName = "DIRECT = OS:ENG-4112-09-
od\private$\cpq"
  Dim cqQueue As MSMQQueue
  Set cqQueue = cqInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
  Dim cqMsg As MSMQMessage
  Set cqMsg = New MSMQMessage
  cqMsg.Label = "B"
  cqMsg.Body = cTime
  cqMsg.Send cqQueue
  cqQueue.Close
End Sub

Private Sub VBA_Block_2_Fire()
  'Resumes the model A
  Dim saqInfo As New MSMQQueueInfo
  Dim saqDest As MSMQQueue
  Dim samsgSend As New MSMQMessage
  samsgSend.Label = "R"
  samsgSend.Body = ThisDocument.Model.SIMAN.RunCurrentTime
  saqInfo.FormatName = "DIRECT = OS:ENG-4112-
07\private$\asq"
  Set saqDest = saqInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
  samsgSend.Send saqDest
  saqDest.Close
End Sub

Private Sub VBA_Block_3_Fire()
  'Resumes the model C
  Dim scqInfo As New MSMQQueueInfo
  Dim scqDest As MSMQQueue
  Dim scmsgSend As New MSMQMessage
  scmsgSend.Label = "R"
  scmsgSend.Body = ThisDocument.Model.SIMAN.RunCurrentTime
  'scqInfo.FormatName = "DIRECT = OS:ENG-4130-
12\private$\csq"
  scqInfo.FormatName = "DIRECT = OS:ENG-4112-09-
od\private$\csq"
  Set scqDest = scqInfo.Open(MQ_SEND_ACCESS, MQ_DENY_NONE)
  scmsgSend.Send scqDest
  scqDest.Close
End Sub
```